Wijmo 5

version 5.20183.550

Table of Contents

Wijmoへようこそ

WijmoはJavaScriptコントロールの新世代を象徴します。 最新の優れたHTML5技術をフル活用し、レガシーブラウザーをサポートするために妥協しません。結果は、素早くて軽い、かつ従来より使いやすいコントロールのセットです。

WijmoはECMAScript5以外に何も依存関係がありません。 jQuery、jQueryUIや他のフレームワークなしでも使用できます。

Wijmoでは、以下の技術を利用するため、モダンブラウザ(Internet Explorer 9や以降)が必要です。

- **ECMAScript 5:** ECMAScript 5規格で、プロパティのgetterとsetterのサポートが追加されました。これは小さな変化のように思われるかもしれませんが、大きな違いを生み出します。たとえば、プロパティの値を増やすときに、 `control.value(control.value() + 1)` と記述するところを `control.value++` と記述できるようになりました。ECMAScript 5規格ではその他にもさまざまな機能拡張が追加されています（たとえば、コールバックで'this'パラメーターの値を指定できる**bind**メソッドなど）。また、新しい**配列**メソッドによって時間を短縮できます。
- **SVG:** 最新のブラウザーでは**SVG**が実装されており、目を見張るようなデータのグラフィック表現を簡単に作成できるようになっています。WijmoはSVGを直接利用します。レガシーブラウザーをサポートする場合に必要であったオーバーヘッドはかかりません。
- **TypeScript:** Wijmoは**TypeScript**で記述されており、型チェックとOOPの概念（モジュール、クラス、継承など）を利用しています。それでも出力は純粋なJavaScriptなので、開発作業にはどちらの言語でも使用できます。
- **モバイルデバイス:** Wijmoは、開発当初からモバイルブラウザーのサポートを前提に設計されました。レスポンシブレイアウトとタッチサポートが、すべてのWijmoコントロールの設計および実装における重要な要素でした。
- **AngularJS：**AngularJSは、現在、最もよく使用されている強力なJavaScriptアプリケーションフレームワークの1つです。Wijmoは、Angular 1.xをリリース以来サポートしており、Angular 2はリリース前からサポートしています。詳細については、「 AngularJSアプリケーションでのWijmoの使用」を参照してください。
- **他のフレームワーク：**Wijmoは、他の任意のJavaScriptフレームワークと共に使用できます。AngularJS に加えて、 **React**、**Vue**、および**KnockoutJS** 用の相互運用モジュールを提供しています。顧客の要望に基づいて、今後は他のフレームワークも追加する予定です。
- **Bootstrap: Bootstrap**は、非常に使いやすく強力で人気が高いCSSフレームワークの1つです。当社のサンプルやオンラインドキュメントではこのCSSフレームワークが使用されています。Bootstrapを使用すれば、追加の労力をかけずにWijmoをサイトのデザインに溶け込ませることができます。

現状では、依然としてレガシーブラウザー（Internet Explorer 8以前）のサポートが必要とされています。そのため、お客様の要望があるかぎり、Wijmo 3も引き続き提供される予定です。IE8以前のサポートが必要な場合は、Wijmo 3を使用し続けてください。旧バージョンのWijmoの保守とサポートは通常どおり継続されます。HTML5および最新ブラウザーに移行する準備ができた場合は、Wijmoをお選びください。

このサイトでは、WijmoのAPIに関する詳細な情報が含まれています。 Wijmoの主な機能や基本的な使用方法については、プレイ**Wijmo**をご確認ください。

アプリケーションでのWijmoの参照

Wijmoをアプリケーションで使用するには、いくつかの参照をHTMLページに追加する必要があります。

Wijmoアプリケーションに必要な最小限のファイルセットは以下のとおりです。

- **wijmo.css:** すべてのWijmoコントロールのスタイル設定に使用されるCSSルールが含まれています。
- **wijmo.js:** Globalize、Event、Control、CollectionViewクラスなどのWijmoインフラストラクチャが含まれています。

これらに加えて、使用するコンポーネントに応じて以下のファイルを1つ以上追加します。

- **wijmo.grid.js:** FlexGridコントロールが含まれています。
- **wijmo.chart.js:** FlexChartとFlexPieコントロールが含まれています。
- **wijmo.input.js:** ComboBox、AutoComplete、InputDate、InputTime、InputNumber、InputMask、ListBox、Menu、Calendarなどの入力コントロールが含まれています。
- **wijmo.gauge.js:** LinearGauge、RadialGauge、BulletGraphなどのゲージコントロールが含まれています。
- **wijmo.nav.js:** TabPanel、TreeViewなどのナビゲーションコントロールが含まれています。
- **wijmo.culture.*[CultureName]*.js:** アメリカ英語以外の言語を対象とするアプリケーションの開発に使用されるカルチャ固有ファイルが含まれています。
- **wijmo.theme.*[ThemeName]*.css:** Wijmoコントロールの外観のカスタマイズに使用されるCSSルールが含まれています。Use any of these files instead of **wijmo.css**.

Wijmoには、最も普及しているJavaScriptフレームワーク用のネイティブコンポーネントとしてコントロールを使用できる相互運用モジュールが含まれています。

- **wijmo.angular.js:** AngularJSアプリケーションでWijmoコントロールをカプセル化するディレクティブが含まれています。
- **wijmo.angular2.js:** Angularアプリケーションでwijmoコントロールをカプセル化するコンポーネントが含まれています。
- **wijmo.react.js:** Reactアプリケーションでwijmoコントロールをカプセル化するコンポーネントが含まれています。
- **wijmo.vue.js:** Vue 1.xアプリケーションでwijmoコントロールをカプセル化するコンポーネントが含まれています。
- **wijmo.vue2.js:** Vue 2.xアプリケーションでwijmoコントロールをカプセル化するコンポーネントが含まれています。
- **wijmo.knockout.js:** Knockoutアプリケーションでwijmoコントロールをカプセル化するコンポーネントが含まれています。

これらのファイルを実際にどこから取得するかについては、2通りの方法があります。1つはWijmoをダウンロードして必要なファイルをアプリケーション内の適切なフォルダーに配置する方法で、もう1つは当社のコンテンツデリバリーネットワーク（CDN）上のクラウドでホストされているWijmoファイルを参照する方法です。以下のセクションでこれらの例を示します。

Wijmoをローカルに配置する

Wijmoファイルをダウンロードし、それらのファイルをアプリケーション内のフォルダーにコピーします。Wijmoスクリプトファイルを"scripts/vendors"というフォルダーに配置し、CSSファイルを"styles"というフォルダーに配置した場合、以下の行をHTMLページに追加できます。

```
<!-- <!-- Wijmo styles and core (必須) -->

<link href="styles/wijmo.min.css" rel="stylesheet"/>
<!--
    optionally use a custom theme instead of wijmo.min.css
    <link href="styles/themes/wijmo.theme.modern.min.css" rel="stylesheet"/>
-->
<script src="scripts/vendor/controls/wijmo.min.js"></script>


<!-- Wijmoコントロール（オプションで、必要なコントロールを含みます） -->
<script src="scripts/vendor/controls/wijmo.grid.min.js"></script>
<script src="scripts/vendor/controls/wijmo.chart.min.js"></script>
<script src="scripts/vendor/controls/wijmo.input.min.js"></script>
<script src="scripts/vendor/controls/wijmo.gauge.min.js"></script>

<!-- Wijmoのカスタムカルチャー（オプション、必要なカルチャーを含みます） -->
<script src="scripts/vendor/controls/cultures/wijmo.culture.ja.min.js "></script>

<!-- AngularJS and Wijmo framework interop (optional, could be Angular/React/Vue/etc) -->
<script src="scripts/vendor/angular.min.js"></script>
<script src="scripts/vendor/interop/angular/wijmo.angular.min.js"></script>

<!-- Wijmoライセンスキーを適用する（オプションです） -->
<script>
    wijmo.setLicenseKey('ここにライセンスキーを入力する...');
</script>
```

WijmoをCDNから取得する

この場合は何もダウンロードする必要はありません。単に以下の行をHTMLページに追加するだけです。

```
<!-- Wijmo styles and core (required) -->
<link href="https://cdn.grapecity.com/wijmo/5.20182.500/styles/wijmo.min.css" rel="stylesheet"/>
<!--
    optionally use a custom theme instead of wijmo.min.css
    <link href="https://cdn.grapecity.com/wijmo/5.20182.500/styles/themes/wijmo.theme.modern.min.css" rel="stylesheet"/>
-->
<script src="https://cdn.grapecity.com/wijmo/5.20182.500/controls/wijmo.min.js"></script>



<!-- Wijmoコントロール（オプションで、必要なコントロールを含みます） -->
<script src="https://cdn.grapecity.com/wijmo/5.20182.500/controls/wijmo.grid.min.js"></script>
<script src="https://cdn.grapecity.com/wijmo/5.20182.500/controls/wijmo.chart.min.js"></script>
<script src="https://cdn.grapecity.com/wijmo/5.20182.500/controls/wijmo.input.min.js"></script>
<script src="https://cdn.grapecity.com/wijmo/5.20182.500/controls/wijmo.gauge.min.js"></script>

<!-- Wijmoのカスタムカルチャー（オプション、必要なカルチャーを含みます） -->
<script src="https://cdn.grapecity.com/wijmo/5.20182.500/controls/cultures/wijmo.culture.ja.min.js "></script>

<!-- AngularJS and Wijmo framework interop (optional, could be Angular/React/Vue/etc)) -->
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular.min.js"></script>
<script src="https://cdn.grapecity.com/wijmo/5.20182.500/interop/angular/wijmo.angular.min.js"></script>

<!-- Wijmoライセンスキーを適用する（オプションです） -->
<script>
    wijmo.setLicenseKey('ここにライセンスキーを入力する...');
</script>
```

最新のリリースバージョンを常に使用する場合は、参照のバージョン番号を「5.latest」で置き換えてください。 例えば：

```
<!-- Wijmoの最新バージョン -->
<script src="https://cdn.grapecity.com/wijmo/5.latest/controls/wijmo.min.js"></script>
<link href="https://cdn.grapecity.com/wijmo/5.latest/styles/wijmo.min.css" rel="stylesheet"/>
```

注意: 参照するファイルの順番が重要です。 最初に**wijmo.js**モジュールを参照し、その後コントロール毎のモジュール およびコントロールの拡張ファイル。**wijmo.angular**などの相互運用モジュールは、最後に参照します。

npmベースのアプリケーションでWijmoを参照する

インストール

Wijmoはnpm上で**wijmo**パッケージとして表現されます。 これを使用するには、通常のnpmパッケージと同様に、 お使いのシステムに**NodeJS**がインストールされている必要があります。

wijmoの最新リリースバージョンを**NodeJS**コマンドプロンプトで次のコマンドを実行してnpmからインストールできます。

```
npm install wijmo
```

リリースビルドに加えて、リリース候補 (RC) 版とナイトリービルドも公開されていますが、 テスト目的のみで作成しているので、生産で使用しないでください。

最新のリリース候補(RC)版ビルドをインストールするには、次のコマンドを実行します。

```
npm install wijmo@rc
```

最新のナイトリービルドをインストールするには、次のコマンドを実行します。

```
npm install wijmo@nightly
```

アプリケーションにWijmoを追加する

Wijmoパッケージには、CommonJSフォーマットのJavaScriptモジュールが含まれています。 WebpackバンドラやSystemJSランタイムのモジュール ローダーのような、 モジュールをロードできるツールと共に使用できます。

WijmoモジュールをTypeScriptまたはBabelで用意されているアプリケーションにインポートするには、 ES2015の import 文を使用します。 import文でのモジュール名は、 **「wijmo/」**接頭辞で始まり、その後にモジュール名が続きます。 たとえば、次のコードでは 「wijmo.grid」モジュールをインポートし、FlexGridコントロールのインスタンスを作成しています。

```
import * as wjcGrid from 'wijmo/wijmo.grid';
let grid = new wjcGrid.FlexGrid('#hostElement');
```

ES5でコードを記述する場合は、CommonJSの**require()**関数を使用してモジュールをインポートできます。 以下に例を示しています。

```
var wjcGrid = require('wijmo/wijmo.grid');
var grid = new wjcGrid.FlexGrid('#hostElement');
```

人気のあるフレームワークでWijmoを使用する

Wijmo npmモジュールをAngular CLI、create-react-app、Vue CLI、Ionic CLIなどの 人気のある開発ツールと共に使用する方法の詳細については、「**Angular & Wijmoクイックスタート**」の記事をご参照ください。

Wijmoアプリケーションのライセンス

Wijmoのトライアル版モジュールを利用して作成したアプリケーションの下部にはウォーターマークが表示されます。このウォーターマークを削除するには、ライセンスキーが必要であり、以下の簡単な手順を実行します。

1. **Wijmoのコピーを購入**し、当社のWebサイトに**シリアル番号を登録します**。
2. アカウントページに移動して**配布用ライセンスキーを取得します**。
3. 次の例にのように、Wijmoの**setLicenseKey**メソッドを使用して、配布用ライセンスキーをアプリケーションに適用します。

スクリプトタグを使用してWijmoを読み込む場合は、次のように**setLicenseKey**メソッドを呼び出します。

```
<!-- Wijmoの参照 (必須) -->
<script src="https://cdn.grapecity.com/wijmo/5.20182.500/controls/wijmo.min.js"></script>
<link href="https://cdn.grapecity.com/wijmo/5.20182.500/styles/wijmo.min.css" rel="stylesheet"/>

<!-- WijmoのlicenseKeyを適用する -->
<script>
    wijmo.setLicenseKey('ここにライセンスキーを入力する...');
</script>
```

モジュールローダーを使用してWijmoをインポートする場合は、Wijmoコアモジュールで**setLicenseKey**メソッドを呼び出します。

```
import * as wjcCore from 'wijmo/wijmo';
wjcCore.setLicenseKey('ここにライセンスキーを入力する...');
```

上記手順でアプリケーションをライセンスでき、ライセンスキーの作成時に指定した任意のドメインにアプリケーションを配置できます。

Wijmoコントロールの作成

すべてのWijmoコントロールは、ページ上でそのコントロールをホストするHTML要素に関連付けられます。コントロールを作成するには、まず **div**要素をページに追加してから、JavaScriptコードを使用してコントロールのインスタンス化とホスト要素へのバインドを行います。

たとえば、次のフィドルは**FlexGrid**と**FlexChart**を作成して小さいデータソースにバインドする方法を示します。

例: コントロールの作成

このフィドルには必要なすべての参照が含まれています（「アプリケーションでのWijmoの参照」トピックを参照）。フィドルのHTML部分で、以下のように'theGrid'および'theChart'という名前の2つの**div**を定義しています。

```
<h1>Hello</h1>

<p>FlexGridコントロール:</p>
<div id="theGrid"></div>

<p>FlexChartコントロール:</p>
<div id="theChart"></div>

<p>今のところはこれだけ</p>
```

フィドルのJavaScript部分は、ドキュメントがロードされたときに実行されます。これは小さいデータセットを作成し、各コントロールを**div**要素にバインドしてからデータセットにバインドします。

```
<script id="scriptInit">
onload = function () {

  // ランダムなデータを生成します。
  var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(','),
      data = [];
  for (var i = 0; i < countries.length; i++) {
    data.push({
      country: countries[i],
      downloads: Math.round(Math.random() * 20000),
      sales: Math.random() * 10000,
      expenses: Math.random() * 5000
    });
  }

  // グリッドを作成してデータを表示します。
  var grid = new wijmo.grid.FlexGrid('#theGrid', {
    itemsSource: data
  });

  // チャートを作成して同じデータを表示します。
    var chart = new wijmo.chart.FlexChart('#theChart', {
      itemsSource: data,
      bindingX: 'country',
      series: [
          { name: 'Sales', binding: 'sales' },
          { name: 'Expenses', binding: 'expenses' },
          { name: 'Downloads', binding: 'downloads', chartType: wijmo.chart.ChartType.LineSymbols } ]
  });
});
</script>
```

コントロールのサイズと位置はホスト要素によって決定されることに注意してください。この例ではCSSを使用してグリッドの高さを"auto"に設定しているので、その内容が収まるように自動的にグリッドのサイズが設定されます。また、スペースに合わせるために多くの項目がある場合 **max-height**の値を設定し、必要に応じてグリッドに自動的にスクロールバーが表示されるようにすることもできます。

ほとんどの場合、ページのレイアウトにはBootstrapなどのCSSフレームワークを使用し、コントロールを他のHTML要素のように正確にレイアウトします。

Wijmoコントロールをホストしている要素への参照を取得するには、コントロールの**hostElement**プロパティを使用します。特定の要素によってホストされているコントロールへの参照を取得するには、**Control.getControl(element)**静的メソッドを使用します。

コントロールのサイズ設定とレイアウトの詳細については、「コントロールのサイズとスタイルの設定」トピックを参照してください。

div要素はすべてのWijmoコントロールのホストとして使用できます。また、 **input**要素はほとんどの入力コントロールのホストとして、 **select**要素は**ListBox**、**ComboBox**、**AutoComplete**、および**Menu**コントロールのホストとして、それぞれ使用できます。

コントロールのサイズとスタイルの設定

Wijmoコントロールは、スタイル設定、外観、およびサイズ設定にCSSを使用します。

そのため、Wijmoコントロールには"width"、"height"、"background"などのプロパティはありません。スタイル設定とレイアウトはCSSを使用して処理されます。これまで**WinForms**や**XAML**などの.NETコントロールを使用していた開発者の方は、最初この点に違和感を覚えるかもしれません。しかし、CSSに慣れると、これが非常に簡単で強力であることがわかります。再利用性の高いほんの数行のCSSコードを使用して、ある特定のコントロール型のすべてのインスタンスのスタイル、または特定のコントロールのスタイルを簡単に設定できます。

コントロールのサイズ設定

コントロールのサイズと位置は、ホスト要素によって通常のHTML/CSSルールに従って決定されます。たとえば、以下のCSSルールは、"grid"クラスを持つ要素について、グリッドの内容がすべて収まるように高さを自動的に計算し、その高さの上限を300ピクセルにすることを指定します。

```
.grid {
    height: auto;
    max-height: 300px;
}
```

次のフィドルは、このルールがグリッドにどのように作用するかを2つの**FlexGrid**コントロールで示しています。最初のグリッドは項目数が少ないので、（通常のHTMLテーブルのように）そのすべての内容が収まるようにサイズ設定されます。2番目のグリッドは項目数が多いので、高さが300ピクセルに制限されてスクロールバーが表示されます。

例: コントロールのサイズ設定

Show me (https://jsfiddle.net/Wijmo5/J4zME)

最初のグリッドは表示する項目の数が5つだけです。これは300ピクセルの高さに収まるので、グリッドにすべての項目が表示されてスクロールバーは付きません。2番目のグリッドには10,000件の項目が含まれます。これは300ピクセルの高さを超えるため、グリッドはスクロール可能になります。

コントロールのスタイル設定

コントロールのスタイル設定はサイズ設定と同じロジックに従います。CSSを使用して、フォント、色、マージン、パディングといったコントロールの任意の部分の視覚的側面をオーバーライドします。

たとえば、次のフィドルは、CSSを使用して**FlexGrid**コントロールの外観を変更する方法を示します。

例: コントロールのスタイル設定

Show me (https://jsfiddle.net/Wijmo5/3L8Cf)

ここでは、グリッドは白黒のプレーンな外観になっています。このような外観にするため、CSSを変更してグリッド**内部**の要素のスタイルを指定しています。Wijmoコントロールの構成要素にはクラス名が割り当てられているので、スタイルを簡単かつ柔軟に設定できます。

たとえば、以下のCSSは、セル要素（"grid"クラスを持つ要素に含まれる"wj-cell"クラスを持つ要素）の外観を罫線なしで白の背景にします。

```
.grid .wj-cell {
    border: none;
    background-color: #fff;
}
```

コード上でスタイル設定

Wijmoコントロールのレイアウトとサイズ設定は、主にCSSで行われますが、コントロールの各面を制御するにはコードが必要となる場合もあります。

たとえば、FlexGridはコントロールを描画するフォントに応じて行の高さを計算します。ただし、行の高さを自由で設定するには、以下のプロパティを設定してCSS上のスタイルをオーバーライドできます。

```
// スクロール領域で行の高さを設定します。
flex.rows.defaultSize = 34;
// 列ヘッダ領域で行の高さを設定します。
flex.columnHeaders.rows.defaultSize = 40;
```

たとえば、次のフィドルは、グリッドの見た目をカスタマイズするにはCSSを使用します。また、FlexGridコントロールを参照するには、initializedイベントが使用されています。この方法は、マークアップでコントロールが作成されている場合に便利です。

例: コード上でスタイル設定

Show me (https://jsfiddle.net/Wijmo5/kzrutyj9)

疑似クラス

CSS疑似クラスは、選択される要素の特別な状態を指定するためにセレクタに追加されるキーワードです。たとえば、 `:hover` は、セレクタによって指定される要素の上にユーザーがマウスポインタを置いたときにスタイルを適用します。

疑似クラスを使用すると、ドキュメントツリーのコンテンツだけでなく、要素にフォーカスがあるかどうか（ `:focus` ）、要素が無効状態かどうか（ `:invalid` ）などの外部要因に関しても要素にスタイルを適用できるので、疑似クラスはフォーム内で重要です。

標準的な疑似クラスの一部は特定の要素にしか適用されず、また要素の祖先に適用されないため、実用性に制限があります。たとえば、入力要素を含むWijmo入力コントロールは多数あります。それらの入力要素がフォーカスを持つと、内部の入力要素は `:focus` 疑似クラスを取得しますが、その要素のコントロールホストは取得しません。

そこで、Wijmoは、いくつかの独自疑似クラスを追加して、効果的なフォームを容易に構築できるようにしています。

- **wj-state-focused**：コントロールにアクティブな要素が**含まれる**場合にコントロールホスト要素に追加されます（ホスト要素がアクティブな要素**である**場合では必ずしもありません）。This is equivalent to the standard **focus-within** pseudo-selector, which is not available in IE/Edge.
- **wj-state-invalid**：コントロールに無効状態の入力要素が含まれる場合にコントロールホスト要素に追加されます。
- **wj-state-empty**：コントロールにコンテンツがない入力要素が含まれる場合にコントロールホスト要素に追加されます。これは、子を持たない要素に適用される `:empty` 疑似クラスとは異なります。
- **wj-state-readonly**：コントロールの**isReadOnly**プロパティがtrueに設定されている場合にコントロールホスト要素に追加されます。
- **wj-state-disabled**：コントロールの**isDisabled**プロパティがtrueに設定されている場合にコントロールホスト要素に追加されます。これは、コントロールのホスト要素に"disabled"属性を追加することに相当します。

以下のサンプルは、直線型ゲージと円形ゲージがフォーカスを取得したときに、**wj-state-focused**疑似クラスを使用してスライダコントロールにCSSアニメーションを適用する例を示します。

例：疑似クラス

Show me (https://jsfiddle.net/Wijmo5/450s0Lym)

WijmoイベントとHTMLイベント

HTML5には、HTML要素に対して機能するイベントメカニズムがありますが、コントロールやコレクションなどの任意のオブジェクトにイベントを追加することはできません。

このため、Wijmoは、すべてのWijmoクラスにあらゆるイベントを実装するために使用できる **Event**クラスを定義しています。WijmoイベントとHTMLイベントの主な相違点は次のとおりです。

1. Wijmoは、（HTML要素だけでなく）任意のクラスで宣言できます。
2. Wijmoイベントはルーティング（キャプチャとバブル）を含まないので、HTMLイベントより軽量です。Wijmoイベントは、それを宣言したオブジェクトだけをターゲットにします。
3. Wijmoイベントハンドラの追加または削除は、（HTMLイベントで使用される **addEventListener**メソッドまたは**removeEventListner**メソッドではなく）イベントの**addHandler**メソッドまたは**removeHandler**メソッドを呼び出して行うことができます。
4. Wijmoのイベントハンドラはそれぞれ、（a）イベントの送信元と（b）イベント引数の2つのパラメータを受け取ります。
5. Wijmoイベント"XYZ"は、対応するメソッド"onXYZ"によって生成されます。ただし、派生クラスでオーバーライドして、ハンドラをアタッチせずにイベントを処理したり、イベントをカスタマイズしたり、抑止することもできます。

Wijmoイベントは、HTMLイベントに代わるものではありません。通常、アプリケーションでは両方を使用します。コントロールの **hostElement**またはコントロールテンプレート内で定義される要素をターゲットにするマウス操作やキーボード操作を処理するには、HTMLイベントを使用します。DOMに直接関連しないコントロール固有のイベント（**valueChanged**や**rowAdded**など）を処理するには、Wijmoイベントを使用します。

以下の例は、プレーンJavaScriptを使用して、 **InputNumber**コントロールのHTMLイベントとWijmoイベントにハンドラを追加する方法を示します。

```
// コントロールを作成します
var ctl = new wijmo.input.InputNumber('#inputNumber');

// Wijmoイベントハンドラをアタッチします
ctl.valueChanged.addHandler(function (s, e) {
    console.log('the value has changed to ' + s.value);
});

// HTMLイベントハンドラをアタッチします
ctl.addEventListener(ctl.hostElement, 'keypress', function(e) {
    console.log('you pressed ' + e.charCode);
});
```

上の例は、プレーンJavaScriptを使用する構文を示します。Angular、Knockout、Aurelia、Vueなどのフレームワークを使用するアプリケーションでは、それぞれのフレームワークで規定された構文を使用する必要があります。

たとえば、Angular 1.xでは、次のようにして**valueChanged** Wijmoイベントにハンドラをアタッチします。

```
<wj-input-number
    value-changed="myValueChangedEventHander(s, e)">...
```

Angular2では、次のようにします。

```
<wj-input-number #theControl
    (value-changed)="myValueChangedEventHander(theControl, $event)">...
```

HTMLイベントとWijmoイベントの詳細については、 **HTMLイベントとWijmoイベント** ブログおよびEventクラスに関するドキュメントを参照してください。

Wijmoでの列挙体の使用

いくつかのWijmoコントロールには列挙値をとるプロパティがあります。

たとえば、**FlexChart**の**chartType**プロパティは**wijmo.chart.ChartType**の値をとります。

列挙プロパティの設定

列挙プロパティの推奨される設定方法は以下のとおりです。

```
// 列挙プロパティの値を設定します。
chart.chartType = wijmo.chart.ChartType.Line;
```

以下のような記述方法でも同じ結果になります。

```
// wijmo.chart.ChartType.Lineには値3が割り当てられています。
chart.chartType = 3;

// 列挙体が自動的に解析されます。
chart.chartType = 'Line';
```

列挙プロパティの取得

上記のプロパティを取得すると、どの場合でも3が返されます。（たとえばUIに表示するために）値を文字列として取得する場合は、以下のように記述できます。

```
// 列挙値を数値として取得します。
console.log(chart.chartType); // "3" を出力します。

// 列挙値を文字列として取得します。
console.log(wijmo.chart.ChartType[chart.chartType]); // "Line" を出力します。
```

列挙値の変換

列挙体クラスとインデックスを使用して、文字列と対応する数値を相互に変換できます。例:

```
// 列挙値を文字列に変換します。
console.log(wijmo.chart.ChartType[3]); // "Line" を出力します。
console.log(wijmo.chart.ChartType[1000]); // "null" を出力します。

// 文字列を列挙値に変換します。
console.log(wijmo.chart.ChartType['Line']); // "3" を出力します。
console.log(wijmo.chart.ChartType['NoSuchValue']); // "null" を出力します。
```

.NET開発者向けのメモ

.NETの**Enum**クラスには、列挙体によって定義された名前と値を返す**GetNames**および**GetValues**というメソッドがあります。

以下のコードは、（Wijmoで使用されている）TypeScript列挙体によって定義された名前と値を取得する同じようなメソッドを実装する方法を示します。

```
// 列挙体によって定義された名前を取得します。
function getEnumNames(enumClass) {
    var names = [];
    for (var key in enumClass) {
        var val = parseInt(key);
        if (isNaN(val)) names.push(key);
    }
    return names;
}

// 列挙体によって定義された値を取得します。
function getEnumValues(enumClass) {
    var values = [];
    for (var key in enumClass) {
        var val = parseInt(key);
        if (!isNaN(val)) values.push(val);
    }
    return values;
}

// 使用例:
var nn = getEnumNames(wijmo.DataType); // [ 'Object', 'String', 'Number', 'Boolean', 'Array' ]を返します。
var vv = getEnumValues(wijmo.DataType); // [ 0, 1, 2, 3, 4 ]を返します。
```

Wijmoアプリケーションのグローバライズ

デフォルトでは、Wijmoはアメリカ英語カルチャを使用してデータの書式設定と解析を行います。小数点記号はピリオド、桁区切り文字はカンマ、曜日は"Sunday"から"Saturday"です。

作成するアプリケーションの対象言語が他のカルチャである場合は、該当するWijmoカルチャファイルへの参照をHTMLページに含めます。Wijmoには40種類ほどのカルチャファイルが付属しており、新しいカルチャを生成するツールもあります。現在サポートされていないカルチャを対象とする場合は、当社にお問い合わせいただければ、必要なファイルを作成してお送りします。

たとえば、アプリケーションをドイツカルチャ用にローカライズするには、以下の行をページのヘッドセクションに追加します。

```
<!-- Germanカルチャを設定します。 -->
<script src="https://cdn.grapecity.com/wijmo/5.latest/controls/cultures/wijmo.culture.de.min.js"></script>
```

次のフィドルはこの結果を示します。グリッドの値とカレンダーの日付の書式を確認してください。

例: グローバリゼーション

Show me (https://jsfiddle.net/Wijmo5/wK97R)

グリッドの値を編集すると、データの解析にもグローバリゼーションが機能していることがわかります。WijmoのGlobalizeクラスを使用すると、アプリケーション内のWijmoコントロール以外の場所でも同様に値の書式設定と解析を行うことができます。

対応するカルチャ

Wijmoではデフォルトとしてアメリカ英語カルチャが設定されていますが、利便性のため他のカルチャもいくつか対応しています。CDNにも搭載されていますが、[インストールフォルダ\Dist\controls\cultures]にも存在します。現在、対応されているカルチャは以下のとおりです。

- **ar-AE** アラビア語（U.A.E.）
- **eu** バスク語
- **bg** ブルガリア語
- **ca** カタロニア語
- **zh** 中国語
- **zh-HK** 中国語（繁体字、香港）
- **zh-TW** 中国語（繁体字、台湾）
- **hr** クロアチア語
- **cs** チェコ語
- **da** デンマーク語
- **nl** オランダ語
- **en** 英語
- **en-CA** 英語（カナダ）
- **en-GB** 英語（英国）
- **et** エストニア語
- **fi** フィンランド語
- **fr** フランス語
- **fr-CA** フランス語（カナダ）
- **gl** ガリシア語
- **de** ドイツ語
- **el** ギリシア語
- **he** ヘブライ語
- **hi** ヒンディー語
- **hu** ハンガリー語
- **id** インドネシア語
- **it** イタリア語
- **ja** 日本語
- **kk** カザフ語
- **ko** 韓国語
- **lv** ラトビア語
- **lt** リトアニア語
- **no** ノルウェー語

- **pl** ポーランド語
- **pt** ポルトガル語
- **ro** ルーマニア語
- **ru** ロシア語
- **sr** セルビア語
- **sk** スロバキア語
- **sl** スロベニア語
- **es** スペイン語
- **es-419** スペイン語 (ラテンアメリカ)
- **es-MX** スペイン語 (メキシコ)
- **sv** スウェーデン語
- **th** タイ語
- **tr** トルコ語
- **uk** ウクライナ語

和暦の表示

Wijmoでは日本カルチャを設定することで和暦の表記をサポートしています。 formatプロパティに"g"を使用して書式設定します。

和暦の利用 (https://jsfiddle.net/Wijmo5_jp/emcsmqhe/)

和暦は日本語カルチャファイルに以下のように定義されています。年号を追加したい場合はこの部分に定義を追加することで対応することができます。

```
eras: [
    // 2019/01/01から「新元」という元号が開始する場合
    { name: '新元', symbol: 'A', start: new Date(2019, 0, 1) },
    { name: '平成', symbol: 'H', start: new Date(1989, 0, 8) },
    { name: '昭和', symbol: 'S', start: new Date(1926, 11, 25) },
    { name: '大正', symbol: 'T', start: new Date(1912, 6, 30) },
    { name: '明治', symbol: 'M', start: new Date(1868, 8, 8) }
],
```

※制限事項 現時点で和暦は表示のみをサポートしています。和暦を使用して日付データを編集することはできません。

Wijmoグリフ

wijmo.cssファイルでは、Pure CSSで定義したグリフが含まれています。 グリフはWijmoコントロールと拡張で使用され、Wijmoで作成する独自の アプリケーションでも使用可能です。

CSSを使用してグリフを定義する場合は、他のフォントまたは画像ファイルを配置する必要がなく、 画像は現在のテーマで定義された色とフォントサイズで描画されます。

アプリケーションでWijmoグリフを使用するには、マークアップにてspan要素を追加してそのクラスをグリフ名に設定します。例えば：

```
<span class="wj-glyph-diamond"></span>
```

CSSを使用して、Wijmoコントロール内で使用するグリフの外観をカスタマイズできます。たとえば、次のCSSを使用すると、 **FlexGrid**で行が編集モードであることを示すために使用される鉛筆グリフを非表示にしたり、その外観を変更することができます。

```
/* FlexGridコントロール内の鉛筆グリフを非表示にします */
.wj-flexgrid .wj-glyph-pencil {
    display: none;
}
```

```
/* FlexGridコントロール内の鉛筆グリフをカスタム画像に置き換えます */
.wj-flexgrid .wj-glyph-pencil {
    background-image:url('../images/my-pencil.png');
    background-repeat: round;
    border: 0;
    opacity: 1;
}
    .wj-flexgrid .wj-glyph-pencil:after {
        display: none;
    }
```

以下の表は、**wijmo.css**ファイルで定義されているグリフの一覧を示しています：

グリフ名	グリフ	マークアップ
asterisk	✱	`<span class="wj-glyph-asterisk"></span>`
calendar		`<span class="wj-glyph-calendar"></span>`
check	✔	`<span class="wj-glyph-check"></span>`
circle	●	`<span class="wj-glyph-circle"></span>`
clock		`<span class="wj-glyph-clock"></span>`
diamond	◆	`<span class="wj-glyph-diamond"></span>`
down	▼	`<span class="wj-glyph-down"></span>`
down-left	◣	`<span class="wj-glyph-down-left"></span>`
down-right	◢	`<span class="wj-glyph-down-right"></span>`
file		`<span class="wj-glyph-file"></span>`
filter		`<span class="wj-glyph-filter"></span>`
left	◀	`<span class="wj-glyph-left"></span>`
minus	▬	`<span class="wj-glyph-minus"></span>`
pencil	✎	`<span class="wj-glyph-pencil"></span>`
plus	✚	`<span class="wj-glyph-plus"></span>`
right	▶	`<span class="wj-glyph-right"></span>`
square	■	`<span class="wj-glyph-square"></span>`
step-backward	⏮	`<span class="wj-glyph-step-backward"></span>`
step-forward	⏭	`<span class="wj-glyph-step-forward"></span>`
up	▲	`<span class="wj-glyph-up"></span>`
up-left	◤	`<span class="wj-glyph-up-left"></span>`
up-right	◥	`<span class="wj-glyph-up-right"></span>`

JavaScript インテリセンス

IntelliSenseは、Visual StudioとVSCodeの最も便利な機能の1つです。タイピング量とミスを減らし、作業しながらオブジェクトモデルについて学んだり覚えることができます。

IntelliSenseは、TypeScriptでは十分に機能しますが、JavaScriptでは多少限定的です。言語機能に対する基本的なサポートは得られますが、カスタム定義ファイルを指定しないと、外部ライブラリはサポートされません。

2016/V2リリースから、Wijmoライブラリ全体に対するIntelliSense定義ファイルの提供を開始しました。これらのファイルをインストールするだけで、Wijmoを使用してJavaScriptコードを記述する際に完全なIntellliSense機能が提供されます。

Visual Studio 2017でのインストール

Visual Studio 2017で、JavaScript インテリセンスをTypeScript宣言 (*.d.ts) ファイルを使用して有効にすることができます。宣言ファイルは、Wijmo配布物の「Dist\Controls」フォルダ内に存在します。Visual Studio 2017のアプリケーションでWijmo JavaScript インテリセンスを有効にすることは、アプリケーションがWijmoをどのように参照しているかによります。

グローバルモジュールを使用する場合（Script タグによる読み込み）：

WijmoモジュールがScriptタグによって読み込まれる場合は、次の図に示すように、必要なTypeScript宣言ファイルをプロジェクトに含める必要があります。

NPMモジュールを使用する場合

WijmoモジュールがNPMによって読み込まれる場合、Visual Studio 2017は「node_modules」フォルダでの宣言ファイルを自動的に使用します。したがって、この場合、特に何もする必要はありません。

Visual Studio 2015でのインストール

Visual Studio 2015以下のバージョンにWijmo IntelliSense定義ファイルをインストールするには、"wijmo.intellisense.js"ファイルが必要です。このファイルは、Wijmo配布物の"IntelliSense"フォルダにあります。

```
C1Wijmo-Enterprise_5.xxxxx.xxx.zip
  Dist
    IntelliSense
      readme.txt
      wijmo.intellisense.js
    Samples
      VSTemplates
```

Visual StudioにWijmo IntelliSense定義ファイルをインストールするには、次の手順に従います。

1. "wijmo.intellisense.js"ファイルをコンピュータにコピーします。
2. 次のように選択して、Visual Studioの設定ペインを開きます。

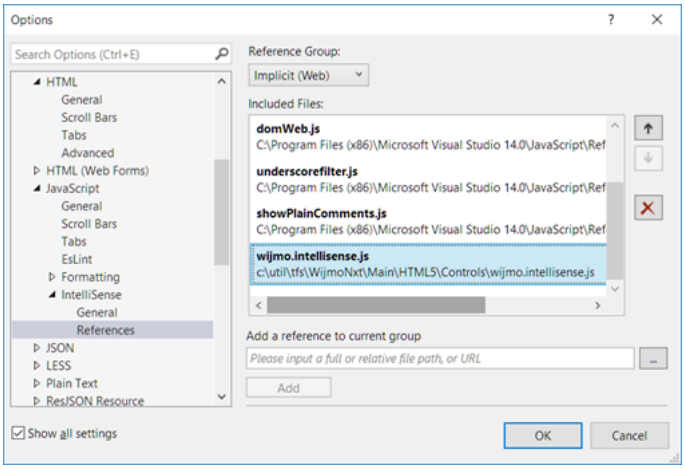
   ```
   ［ツール］→［オプション］→［テキストエディタ］→［JavaScript］→［IntelliSense］→［参照］
   ```

3. ［参照グループ］コンボボックスで、以下を選択します。

   ```
   Implicit (Web)
   ```

4. "wijmo.intellisense.js"ファイルへの参照を追加します。
5. ［OK］をクリックして変更を適用します。

次の図に、このダイアログを示します。



VSCodeでのインストール

VS CodeにWijmo IntelliSense定義をインストールするには、TypeScript定義ファイル（*.d.ts）が必要です。これらのファイルは、Wijmo配布物の"Controls"フォルダにあります。

```
C1Wijmo-Enterprise_5.xxxxx.xxx.zip
  Dist
    controls
      wijmo.d.ts
      wijmo.input.d.ts
      wijmo.grid.d.ts
      …
  IntelliSense
        Samples
        VSTemplates
```

"*.d.ts"ファイルをVSCodeプロジェクトルート下のフォルダにコピーします。プロジェクトに"jsconfig.json"構成ファイルがある場合は、"*.d.ts"ファイルが含まれていることを確認します。次に例を示します。

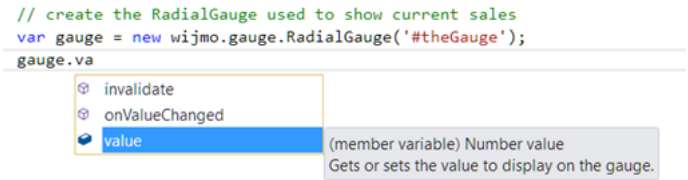
```
"files": [
  "typings/wijmo.d.ts",
  "typings/wijmo.input.d.ts",
  "typings/wijmo.grid.d.ts",
  …
]
```

あるいは、"js"ファイルの先頭に参照コメントを追加して、"*.d.ts"ファイルを指定することもできます。次に例を示します。

```
/// <reference path="typings/wijmo.d.ts" />
```

Wijmo IntelliSenseの使用

Wijmo IntelliSense定義ファイルがインストールされると、JavaScriptコードを記述する際に、Wijmoクラスのオブジェクトモデルを検索したり調べることができます。次の図に、例を示します。



AngularJSディレクティブ

ComponentOneでは、GoogleのJavaScriptアプリケーション用フレームワークである**AngularJS**を支持しています。AngularJSはテンプレート機能、データバインディング、MVVM、Webコンポーネントなどを備えています。

AngularJSの主な利点の1つはMVVMパターンをサポートしていることです。MVVMパターンでは、アプリケーションロジックはモデル（JavaScriptではコントローラーと呼びます）に格納され、外観はビュー（HTML）に格納されます。

これを実現するため、AngularJSでは**ディレクティブ**（カスタムのHTML要素と属性）がサポートされています。最初から多くのディレクティブが組み込まれているほか、WinFormsやXAMLアプリケーションで独自のコントロールを作成するのと同じように独自のディレクティブを簡単に実装できます。

WijmoではすべてのコントロールにAngularJSディレクティブが用意されています。これらのディレクティブは**wijmo.angular.js**ファイルで定義されており、以下のように記述できます。

```
<div ng-app="app" ng-controller="appCtrl">

<p><b>FlexGrid</b>コントロール:</p>
<wj-flex-grid items-source="data">
    <wj-flex-grid-column header="Country" binding="country"></wj-flex-grid-column>
    <wj-flex-grid-column header="Sales" binding="sales"></wj-flex-grid-column>
    <wj-flex-grid-column header="Expenses" binding="expenses"></wj-flex-grid-column>
    <wj-flex-grid-column header="Downloads" binding="downloads"></wj-flex-grid-column>
</wj-flex-grid>
</div>
```

AngularJSアプリケーションでのWijmoの使用

WijmoのAngularディレクティブをプロジェクトで使用する場合は、アプリケーションでのWijmoの参照トピックで説明されているように、まずjQueryへの参照を追加してから、さらにAngularJS、Wijmo、およびWijmoのAngularJSディレクティブへの参照を追加します。これらの参照を適切な位置に記述した後、次のようなコードを使用してappが"wj"モジュールに依存することをAngularJSに伝えます。

```
var app = angular.module('app', ['wj']);
```

appを定義したら、AngularJSアプリケーションの場合と同様にデータとロジックを提供するコントローラーを追加できます。

すべてのWijmoディレクティブは"wj"で始まり、その後に（キャメルケースではなくダッシュを使用して）コントロールのクラス名が続きます。ディレクティブはコントロールのプロパティと一致する属性を持っており、これらも同じ命名規則に従います。

一部のディレクティブは入れ子になったサブディレクティブをサポートしています。たとえば、**wj-flex-grid**ディレクティブには1つ以上の**wj-flex-grid-column**ディレクティブを含めることができ、**wj-flex-chart**ディレクティブには1つ以上の**wj-flex-chart-series**ディレクティブを含めることができます。そのため、マークアップ構文がXAMLによく似たリッチで表現力豊かなものになります。この柔軟性は、MVVMの真の利点を実現するために不可欠です。コントロールの外観とレイアウトはビューによって定義されます。コントローラーはデータを提供するだけです。

次のフィドルはこの仕組みを示します。

例 : Angularディレクティブ

Show me (https://jsfiddle.net/Wijmo5/QNb9X)

WijmoとAngularJSの組み合わせを使用したさまざまな例については、「デモ」を参照してください。

Angular 2コンポーネント

- Angularバージョン**2.2.1**以降（**6.***バージョンを含む）をサポートしています。

メモ 以下の説明は、ビルド211より後に導入された新しい*外部*Wijmoコアライブラリモジュールに関連する事項です。これは、Angular 2用のWijmo相互運用を使用する場合にお勧めする方法です。*グローバル*Wijmoコアライブラリモジュールに基づく説明は、ここにあります。

Wijmo Angular 2の*コンポーネント*により、Angular 2テンプレートマークアップでWijmo *コントロール*を使用できるようになります。TypeScriptクラス継承機能の用語で言えば、Wijmo Angular 2*コンポーネント*は、それらが表す*コントロール*クラスを「拡張」しています。すなわち、Wijmoコンポーネントへの参照を取得した場合、参照されるインスタンスは、コンポーネントのすべてのプロパティ、イベント、メソッドを含むWijmo *コントロール*であると同時に、Angular 2*コンポーネント*でもあります。Wijmoコンポーネントは、コントロールクラスを拡張し、Angular 2テンプレートマークアップでコントロールを使用できるようにするために必要な機能を追加し、完全に機能する一方向/双方向のプロパティ連結およびイベント連結を備えています。Wijmoコントロール、Wijmo Angular 2コンポーネント、そしてAngular 2自体がすべて同じTypeScript言語で記述されているため、この統合はスムーズです。

Wijmo Angular 2コンポーネントは、名前に「angular2」を含む一連のモジュール（1つのコアライブラリモジュールに対して1つのモジュール）として出荷されています。 たとえば、「wijmo.angular2.grid.js」モジュールは、「wijmo.grid.js」コアモジュールに含まれるコントロールに対応するコンポーネントを表します。モジュールは**wijmo** npmパッケージに含まれています。インストール方法については、「NpmからWijmoの参照」トピックを参照してください。

すべてのWijmoモジュールは、 *アンビエント*名を使用してインポートする必要があります。これは、先頭に「wijmo/」が付き、「.js」拡張子を省いたモジュール名です。たとえば、次のimport文は「wijmo.angular2.grid.js」モジュールのコンテンツをインポートします。

```
import * as wjGrid from 'wijmo/wijmo.angular2.grid';
```

Wijmoコンポーネントのインポート

この設定により、Wijmo Angular 2モジュールをインポートし、モジュールに含まれるコンポーネントやディレクティブを使用できます。たとえば、次のコードは、MyCmpコンポーネントのテンプレートにWjFlexGridコンポーネントを追加し、追加したグリッドへの参照を**flex**プロパティに設定します。

```
import { Component, NgModule, ViewChild } from '@angular/core';
import { WjGridModule, WjFlexGrid } from 'wijmo/wijmo.angular2.grid';

@Component({
    template: '<wj-flex-grid #flex [itemsSource]="data"></wj-flex-grid>',
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
    @ViewChild('flex') flex: WjFlexGrid;

}

@NgModule({
    imports: [WjGridModule, BrowserModule],
    declarations: [MyCmp]
})
export class MyModule {
}
```

各Wijmo for Angular 2 JavaScriptモジュールには、そのモジュールのすべてのコンポーネントをエクスポートするAngular 2 NgModuleが含まれます。Wijmo NgModuleデコレータの**imports**メタデータプロパティにWijmo NgModuleへの参照を追加するだけで、NgModuleコンポーネントのテンプレートでこれらのコンポーネントを使用できます。

NgModuleの名前は、JavaScriptモジュール名と次のスキーマを使用して構成されます。
Wj<*wijmo.angular2*プリフィックスを省いたJSモジュール名>**Module**

たとえば、**wijmo.angular2.input** JavaScriptモジュールの場合は**WjInputModule** NgModule、**wijmo.angular2.grid.filter** JavaScriptモジュールの場合は**WjGridFilterModule** NgModuleになります。

コードでのWijmoコントロールの作成

Angular 2用のWijmo*コンポーネント*は、テンプレートマークアップで使用されます。コードでWijmoコントロールを作成する場合は、コンポーネントではなく、その目的で用意されているコアモジュールのWijmo *コントロール*を使用する必要があります。コアモジュールの名前は、対応するAngular 2相互運用モジュール名から「angular2」を省略した名前と同じです。たとえば、次のコードは、コードでFlexGridコントロールを作成します。

```
import { FlexGrid } from 'wijmo/wijmo.grid';
let flex = new FlexGrid('#host_element');
```

WjFlexGridコンポーネントではなくFlexGridコントロールを、「wijmo/wijmo.angular2.grid」ではなく「wijmo/wijmo.grid」モジュールからインポートします。

さまざまなローダー/バンドラーツールへの適合

最もよく使用されているモジュールローダーとモジュールバンドラーにWijmoがよく適合していることを確認します。

WebPack

ここで必要な手順は、バンドルにWijmo CSS（WijmoダウンロードzipにあるDist/styles/wijmo.min.cssファイル）をインクルードすることだけです。このファイルをアプリケーションのルートの任意の場所に置き、アプリケーションのCSSに使用する標準的なWebPackの方法を使用してインクルードする必要があります。

```
import 'style!css!wijmo/styles/wijmo.css';
```

style-loaderとcss-loaderをアプリケーションのpackage.jsonファイルの「devDependencies」オプションに追加する必要があります。

```
"devDependencies": {
    "css-loader": "^0.23.1",
    "style-loader": "^0.13.1",
    ... another libraries
}
```

SystemJSローダー

Wijmoアンビエントモジュール名をnode_modulesフォルダ内のWijmo .jsファイルにマップする必要があります。それには、次の設定オプションを追加してSystem.configメソッド呼び出しに渡します。

```
map: {
    'wijmo': 'node_modules/wijmo'
},
packages: {
    'wijmo': {
        defaultExtension: 'js'
    }
}
```

Angular CLI

詳細な指示については、**「Angular & Wijmoクイックスタート - アプリケーションの作成」**ブログを参照してください。

Angular 2マークアップ構文

詳細については、「Angular 2マークアップ構文」トピックを参照してください。

Wijmo and Sass

We recently switched to **Sass** to make it easier to create and maintain Wijmo CSS files.

Sass improves CSS the same way TypeScript improves JavaScript. It provides modularization, variables, structure, compile time syntax checking, selector nesting, functions, and more. Over the last few years, Sass has become the most popular CSS preprocessor, being supported by all popular JavaScript frameworks and bundler tools.

Just like TypeScript, Sass is optional. You can use the CSS files we build with Sass and forget it exists. But if you want to customize the Wijmo styles, create your own themes, or create smaller CSS files that include only the Wijmo modules you use, then we suggest you use Sass.

If you choose to use Sass, you will a Sass tool. Our favorites are:

- **Visual Studio Web Compiler** and
- **Studio Code CSS, Sass, and Less**.

Folder Structure

We use the following folder structure for Wijmo's Sass files:

```
wijmo.scss                  // main scss file
misc
    _constants.scss         // color names
    _glyphs.scss            // Wijmo glyphs
    _mixins.scss            // utility functions
    _variables.scss         // theme variables
core                        // modules in Wijmo core
    _chart.scss
    _chart_hierarchical.scss
    _chart_interaction.scss
    _core.scss
    _gauge.scss
    _grid.scss
    _grid_filter.scss
    _grid_grouppanel.scss
    _input.scss
    _nav.scss
enterprise                  // modules in Wijmo enterprise
    _chart_finance.scss
    _multirow.scss
    _olap.scss
    _sheet.scss
    _viewer.scss
themes                      // bundled themes
    wijmo.theme.cerulean.scss
    wijmo.theme.cleandark.scss
    wijmo.theme.cleanlight.scss
    // etc
```

Following Sass conventions, files with names that start with an underscore are included by other files and do not generate stand-alone CSS files.

The **wijmo.scss** file is used to generate our main CSS file, **wijmo.css**. It contains the following:

```
// wijmo.scss

// misc
@import "misc/mixins"; // functions
@import "misc/constants"; // named colors
@import "misc/variables"; // theme variables
@import "misc/glyphs"; // wijmo glyphs

// core modules
@import "core/core";
@import "core/input";
@import "core/grid";
@import "core/grid_grouppanel";
@import "core/grid_filter";
@import "core/chart";
@import "core/chart_interaction";
@import "core/chart_hierarchical";
@import "core/gauge";
@import "core/nav";

// enterprise modules
@import "enterprise/sheet";
@import "enterprise/multirow";
@import "enterprise/olap";
@import "enterprise/viewer";
@import "enterprise/chart_finance";
```

If you run this through the Web Compiler or any other Sass tool, you will get the regular **wijmo.css** and **wijmo.min.css** output.

Smaller CSS files

If you don't use any of the Wijmo enterprise modules and would like to reduce the size of your CSS files, you could create a **wijmo-core.scss** file that looks like the **wijmo.scss** above minus the last five lines (the ones that include the enterprise modules).

This will create a **wijmo-core.css** file that is about half the size of the full **wijmo.css**.

We have done this for you already. The **wijmo-core.css** file is included in our distribution. But you can use this approach to generate optimized CSS files that include only the modules your application needs. For example, you could create a CSS file that contains rules only for the **input** and **grid** modules.

Custom Themes

In addition to the standard wijmo.css file, Wijmo includes many custom themes. Most themes are created by overriding the value of a few Sass variables. For example, the cerulean theme is created from the **wijmo.theme.cerulean.scss** file:

```
// wijmo.theme.cerulean.scss -- cerulean theme
$wj-bkg: #fcfcfc;
$wj-txt: #082d43;
$wj-bdr: none;
$wj-hdr-bkg: #033c73;
$wj-hdr-txt: #fff;
$wj-sel-bkg: #2a9fd6;
$wj-msel-bkg: #77afc9;
$wj-cell-grp-bkg: #777777;
//? etc ?

@import "../wijmo.scss";
```

The theme file sets some Wijmo variables and then includes the base **wijmo.scss** file. That's all there is to it. The output is a **wijmo.theme.cerulean.css** file that can be used instead of the standard **wijmo.css**.

NOTE: The theme files are self-contained. They replace rather than extend the standard **wijmo.css** file. This is a change from previous versions of Wijmo.

Wijmo デザイナを使用してコントロールをカスタマイズする

Wijmoデザイナは、Wijmoコントロールを視覚的にデザインする機能を提供します。デザイン画面を使用すると、コントロールのインスタンスを作成してプロパティとイベントを設定し、HTML／JavaScriptを生成してアプリケーションに組み込むことができます。また、Wijmoデザイナは、グリッドやチャートなどの複雑なオブジェクトのプロパティを探索することにも役立ちます。

Wijmoデザイナを起動する

デザイナの起動時には、デザイン画面にいくつかのサンプルデータを含む **FlexGrid**コントロールが配置されています。デザイン画面のコントロールは、 デ*ザインモード*で表示されます。つまり、スクロール、サイズ変更などコントロールを操作することはできません。コントロールのオブジェクトモデルを設定するには、ページの右側にある**プロパティ**ペインを使用します。使用可能な各プロパティにはプロパティの型に応じた適切な種類のエディタが表示され、そこで加えた変更は選択したコントロールにすぐに適用されます。

イベント ペインも右側に表示され、選択したコントロールが公開する各イベントを切り替えるトグルボタンが表示されます。デザイナは、オンになっているイベントに対してイベント処理コードを追加します。

デザイン画面では、複数のコントロールが順番に並べて配置されます。ページの上部にある**編集**ツールバーを使用して、デザイン画面のコントロールを移動、複製、または削除します。

コマンド	説明
上へ移動	選択したコントロールをデザイン画面の1つ上の位置に移動します。
下へ移動	選択したコントロールをデザイン画面の1つ下の位置に移動します。
複製	選択したコントロールのコピーを作成し、デザイン画面の元のコントロールの直後にに挿入します。
削除	選択したコントロールをデザイン画面から完全に削除します。

選択したコントロールインスタンスの名前が**編集**ツールバータブの左側に表示されます。

デザイナのメインメニューは初期状態では折りたたまれ、垂直方向のツールバーとして表示されます。 ページの左上隅にあるWijmoロゴをクリックすると、次のコマンドを含むメニューを展開します。

コマンド	説明
開く	**保存**コマンドで保存したJSONファイルを読み込み、その内容をデザイン画面に表示します。
保存	デザイン画面のコンテンツをJSONファイルとしてダウンロードし、 **開く**コマンドを使用して後で再読み込みします。
ツールボックス	Wijmoコントロールの折りたたみ可能なパネルをモジュール名 (`grid` 、 `chart` 、 `input` 、 `gauge` 、 `nav` , `olap`)でグループ化して開きます。コントロール名をクリックすると、コントロールをデザイン画面に追加します。
テーマ	利用可能なWijmoテーマのリストを開きます。テーマ名をクリックすると、デザイン画面のすべてのコントロールにテーマを適用します。
ソースビュー	デザインビューからソースビューに切り替えます。
ヘルプ	このページを表示します。
Wijmoデザイナについて	Wijmoのバージョン番号と著作権表記を表示します。

保存および**開く**コマンドを使用して、デザイナの現在の状態をJSONファイルとして保存し、後でその内容をデザイナに再読み込みして戻すことができます。

デザイナーのコンテンツをHTML／JavaScriptコードとして表示するには、 **ソースビュー**コマンドを使用します。ソースビューでは、WijmoスクリプトとCSSの参照、デザイン画面上の各コントロールの `<div>` タグ、および各コントロールを作成して初期化するスクリプトを含むコードが表示されます。ページからコードスニペットを直接コピーしたり、メインメニューの **保存** コマンドを使用してHTMLファイルを保存したりすることができます。

コマンド	説明
保存	デザインビューで指定されたコントロールの設定内容に対応するスクリプトコードとWijmo参照を含むHTMLファイルとして、デザイン画面のコンテンツをダウンロードします。
デザインビュー	ソースビューからデザインビューに切り替えます。

デザイン画面に戻るには、**デザインビュー**コマンドを使用します。

再配布可能ファイル

お客様は、お客様が本製品を用いて開発したアプリケーションに、本製品の再配布可能ファイルを組み込み、配布することができます。本製品の再配布可能ファイルは、以下のファイルです。

- Distフォルダに含まれる全てのファイル

※注記

- 上記以外のファイルを配布することはできません。
- 再配布可能ファイルの配布は、お客様が本製品を用いて開発したアプリケーションとともに行われる場合に限り許諾されます。再配布可能ファイルのみを単独で配布することはできません。

wijmo モジュール

ファイル `wijmo.js`
モジュール **wijmo**

すべてのコントロールおよびモジュールによって使用されるユーティリティと **Control** および**Event** クラスが含まれます。

クラス

- Binding
- CancelEventArgs
- Clipboard
- Color
- Control
- DateTime
- Event
- EventArgs
- Globalize
- Point
- PrintDocument
- PropertyChangedEventArgs
- Rect
- RequestErrorEventArgs
- Size
- Tooltip
- TooltipEventArgs

インターフェイス

- IEventHandler
- IQueryInterface

列挙体

- Aggregate
- DataType
- Key

プロパティ

- culture

メソッド

- addClass
- animate
- asArray
- asBoolean
- asCollectionView
- asDate
- asEnum
- asFunction
- asInt
- asNumber
- assert

- assert
- asString
- asType
- changeType
- clamp
- closest
- closestClass
- contains
- copy
- createElement
- disableAutoComplete
- enable
- escapeHtml
- format
- getActiveElement
- getAggregate
- getElement
- getElementRect
- getType
- getUniqueId
- getVersion
- hasClass
- hasItems
- hidePopup
- httpRequest
- isArray
- isBoolean
- isDate
- isEmpty
- isFunction
- isInt
- isNullOrWhiteSpace
- isNumber
- isObject
- isPrimitive
- isString
- isUndefined
- mouseToPage
- moveFocus
- removeChild
- removeClass
- setAriaLabel
- setAttribute
- setCss
- setLicenseKey
- setSelectionRange
- setText
- showPopup
- toFixed

- toggleClass
- toHeaderCase
- tryCast

プロパティ

- culture

Wijmoライブラリ内のすべてのローカライズ可能な文字列を含むオブジェクトを取得または設定します。

カルチャセレクターは、**ISO 639カルチャ**を表す2文字の文字列です。

型 **any**

メソッド

- addClass

```
addClass(e: Element, className: string): void
```

要素にクラスを追加します。

パラメーター

- **e**: **Element**
  クラスを追加する要素。
- **className**: **string**
  要素に追加するクラス（またはスペースで区切られたクラスのリスト）。

戻り値 **void**

animate

```
animate(apply: Function, duration?: number, step?: number): any
```

パラメーターが0から1まで変化するタイマーを使用して関数を呼び出します。

この関数を使用すると、ドキュメントのプロパティまたはスタイルをタイマーで変更することによってアニメーションを作成できます。

たとえば、以下のコードは、要素の不透明度を1秒間で0から1に変更します。

```
var element = document.getElementById('someElement');
animate(function(pct) {
  element.style.opacity = pct;
}, 1000);
```

この関数は、アニメーションの停止に使用できる内部IDを返します。アニメーションの停止は通常、新しいアニメーションを開始するときや、同じ要素で現在実行されている他のアニメーションを中止するときに行います。 たとえば、以下のコードは、内部IDを追跡して新しいアニメーションを開始する前にIDをクリアします。

```
var element = document.getElementById('someElement');
if (this._animInterval) {
  clearInterval(this._animInterval);
}
var self = this;
self._animInterval = animate(function(pct) {
  element.style.opacity = pct;
  if (pct == 1) {
    self._animInterval = null;
  }
}, 1000);
```

パラメーター

- **apply**: **Function**
  ドキュメントを変更するコールバック関数。 この関数は、パーセンテージを表す1つのパラメーターをとります。
- **duration**: **number** OPTIONAL
  アニメーションの持続時間（ミリ秒単位）。
- **step**: **number** OPTIONAL
  アニメーションフレームの間隔（ミリ秒単位）。

戻り値 **any**

asArray

```
asArray(value: any, nullOK?: boolean): any[]
```

値が配列であることを表明します。

パラメーター

- **value**: **any**
  配列であるはずの値。
- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。

戻り値 **any[]**

asBoolean

```
asBoolean(value: boolean, nullOK?: boolean): boolean
```

値がブール値であることを表明します。

パラメーター

- **value**: **boolean**
  ブール値であるはずの値。

- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。

戻り値 **boolean**

asCollectionView

```
asCollectionView(value: any, nullOK?: boolean): ICollectionView
```

値が**ICollectionView** または配列であることを表明します。

パラメーター

- **value**: **any**
  配列または**ICollectionView**。

- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。

戻り値 **ICollectionView**

asDate

```
asDate(value: Date, nullOK?: boolean): Date
```

値が日付であることを表明します。

パラメーター

- **value**: **Date**
  日付であるはずの値。

- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。

戻り値 **Date**

asEnum

```
asEnum(value: number, enumType: any, nullOK?: boolean): number
```

値が列挙体の有効な設定であることを表明します。

パラメーター

- **value**: **number**
  列挙体のメンバであるはずの値。
- **enumType**: **any**
  テストする列挙体。
- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。

戻り値 **number**

asFunction

```
asFunction(value: any, nullOK?: boolean): Function
```

値が関数であることを表明します。

パラメーター

- **value**: **any**
  関数であるはずの値。
- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。

戻り値 **Function**

asInt

```
asInt(value: number, nullOK?: boolean, positive?: boolean): number
```

値が整数であることを表明します。

パラメーター

- **value**: **number**
  整数であるはずの値。
- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。
- **positive**: **boolean** OPTIONAL
  正の整数のみを受け入れるかどうか。

戻り値 **number**

asNumber

```
asNumber(value: number, nullOK?: boolean, positive?: boolean): number
```

値が数値であることを表明します。

パラメーター

- **value**: **number**
  数値であるはずの値。
- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。
- **positive**: **boolean** OPTIONAL
  正の数値のみを受け入れるかどうか。

戻り値 **number**

assert

```
assert(condition: boolean, msg: string): void
```

条件が偽の場合、例外をスローします。

パラメーター

- **condition**: **boolean**
  真であることが期待される条件。
- **msg**: **string**
  条件が真でない場合の例外メッセージ。

戻り値 **void**

asString

```
asString(value: string, nullOK?: boolean): string
```

値が文字列であることを表明します。

パラメーター

- **value**: **string**
  文字列であるはずの値。
- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。

戻り値 **string**

asType

```
asType(value: any, type: any, nullOK?: boolean): any
```

値が指定した型のインスタンスであることを表明します。

パラメーター

- **value**: **any**
  チェックする値。
- **type**: **any**
  期待される値の型。
- **nullOK**: **boolean** OPTIONAL
  nullが許容されるかどうか。

戻り値 **any**

changeType

```
changeType(value: any, type: DataType, format: string): any
```

値の型を変更します。

変換が失敗した場合は元の値が返されます。変換が成功したかどうかを確認するには、戻り値の型をチェックします。

パラメーター

- **value**: **any**
  変換する値。
- **type**: **DataType**
  値の変換先の**DataType**。
- **format**: **string**
  文字列に変換、または文字列から変換するときに使用する書式。

戻り値 **any**

clamp

```
clamp(value: number, min: number, max: number): number
```

最小値から最大値までの範囲に値をクランプします。

パラメーター

- **value**: **number**
  元の値。
- **min**: **number**
  許容される最小値。
- **max**: **number**
  許容される最大値。

戻り値 **number**

closest

```
closest(e: any, selector: string): Node
```

セレクターを満たす最も近い祖先（元の要素を含む）を検索します。

パラメーター

- **e**: **any**
  検索を開始する要素。

- **selector**: **string**
  セレクター式を含む文字列に対する要素を一致させます。

戻り値 **Node**

closestClass

```
closestClass(e: any, className: string): Node
```

クラスセレクターを満たす最も近い祖先（元の要素を含む）を検索します。

パラメーター

- **e**: **any**
  検索を開始する要素。

- **className**: **string**
  クラス名を含む文字列に対する要素を一致させます。

戻り値 **Node**

contains

```
contains(parent: any, child: any): boolean
```

HTML要素に別の要素が含まれているかどうかをチェックします。

パラメーター

- **parent**: **any**
  親要素。

- **child**: **any**
  子要素。

戻り値 **boolean**

copy

```
copy(dst: any, src: any): void
```

オブジェクトのプロパティを別のオブジェクトにコピーします。

このメソッドは、通常、コントロールや他のWijmoオブジェクトのプロパティを設定し、イベントハンドラを割り当てることによって、コントロールや他のWijmoオブジェクトを初期化するために使用されます。

コピー先オブジェクトでは、コピー元で定義されているすべてのプロパティが定義されている必要があります。そうでなければ、エラーがスローされます。

パラメーター

- **dst**: **any**
  コピー先オブジェクト。
- **src**: **any**
  コピー元オブジェクト。

戻り値 **void**

createElement

```
createElement(html: string, appendTo?: HTMLElement): HTMLElement
```

HTML文字列から要素を作成します。

パラメーター

- **html**: **string**
  HTMLElementに変換するHTMLフラグメント。
- **appendTo**: **HTMLElement** OPTIONAL
  新しい要素を付加する先のオプションのHTMLElement。

戻り値 **HTMLElement**

disableAutoComplete

```
disableAutoComplete(e: HTMLInputElement): void
```

入力要素のautocompleteプロパティ、autocorrectプロパティ、autocapitalizeプロパティ、およびspellcheckプロパティを無効にします。

パラメーター

- **e**: **HTMLInputElement**
  入力要素。

戻り値 **void**

enable

```
enable(e: HTMLElement, value: boolean): void
```

要素を有効または無効にします。

パラメーター

- **e**: **HTMLElement**
  有効または無効にする要素。

- **value**: **boolean**
  要素を有効にするか無効にするか。

戻り値 **void**

escapeHtml

```
escapeHtml(text: string): void
```

HTML文字をテキストエンティティに置き換えることによって文字列をエスケープします。

ユーザーが入力した文字列は常に、HTMLページに表示する前にエスケープする必要があります。これにより、ページの完全性が維持され、HTML/JavaScriptインジェクション攻撃を防ぐことができます。

パラメーター

- **text**: **string**
  エスケープするテキスト。

戻り値 **void**

format

```
format(format: string, data: any, formatFunction?: Function): string
```

指定された文字列内の各書式項目をオブジェクトの値に対応するテキストに置き換えます。

この関数は、**formatString**内のパターン'{name:format}'を **data**パラメータのプロパティに置き換えます。次に例を示します。

```
var data = { name: 'Joe', amount: 123456 };
var msg = wijmo.format('Hello {name}, you won {amount:n2}!', data);
```

format 関数は複数形式をサポートします。書式文字列が 'count'プロパティと'when'プロパティを含むJSONエンコードオブジェクトの場合、 このメソッドはデータオブジェクトの'count'パラメータを使用して、 'when'プロパティから適切な書式を選択します。次に例を示します。

```
var fmt = {
    count: 'count',
    when: {
        0: 'No items selected.',
        1: 'One item is selected.',
        2: 'A pair is selected.',
        'other': '{count:n0} items are selected.'
    }
}
fmt = JSON.stringify(fmt);
console.log(wijmo.format(fmt, { count: 0 })); // 「No items selected.」
console.log(wijmo.format(fmt, { count: 1 })); // One item is selected.
console.log(wijmo.format(fmt, { count: 2 })); // A pair is selected.
console.log(wijmo.format(fmt, { count: 12 })); 12 items are selected.
```

オプションの**formatFunction**を使用すると、状況依存の書式設定を指定して コンテンツをカスタマイズできます。指定すると、書式要素ごとに書式関数が呼び出され、 データオブジェクト、パラメータ名、書式、および値が渡されて、 出力文字列が返されます。 次に例を示します。

```
var data = { name: 'Joe', amount: 123456 };
var msg = wijmo.format('Hello {name}, you won {amount:n2}!', data,
    function (data, name, fmt, val) {
        if (wijmo.isString(data[name])) {
            val = wijmo.escapeHtml(data[name]);
        }
        return val;
    }
);
```

パラメーター

- **format**: **string**
  各種要素から成る複合的な書式文字列。
- **data**: **any**
  文字列の構築に使用するデータオブジェクト。
- **formatFunction**: **Function** OPTIONAL
  （オプション）コンテキストに応じて項目を書式設定する場合に使用する関数。

戻り値 **string**

getActiveElement

```
getActiveElement(): void
```

シャドウドキュメントフラグメントも考慮して、フォーカスを持つ要素への 参照を取得します。

戻り値 **void**

getAggregate

```
getAggregate(aggType: Aggregate, items: any[], binding?: string): void
```

配列内の値から集計値を計算します。

パラメーター

- **aggType**: **Aggregate**
  計算する集計のタイプ。
- **items**: **any**[]
  集計する項目を含む配列。
- **binding**: **string** OPTIONAL
  集計するプロパティの名前（項目が単純な値でない場合）。

戻り値 **void**

getElement

```
getElement(selector: any): HTMLElement
```

jQueryのセレクターから要素を取得します。

パラメーター

- **selector**: **any**
  要素、セレクター文字列、またはjQueryオブジェクト。

戻り値 **HTMLElement**

getElementRect

```
getElementRect(e: Element): Rect
```

要素の外接矩形（ページ座標単位）を取得します。

これは **getBoundingClientRect** 関数に似ていますが、 ビューポート座標を使用する点が異なります（ビューポート座標はドキュメントのスクロールに従って変更されます）。

パラメーター

- **e**: **Element**

戻り値 **Rect**

getType

```
getType(value: any): DataType
```

値の型を取得します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **DataType**

getUniqueId

```
getUniqueId(baseId: string): string
```

指定されたベースIDに連続する数値を付加して、要素の新しい一意IDを 作成します。

パラメーター

- **baseId**: **string**
  一意IDを生成するためのベースとして使用する文字列。

戻り値 **string**

getVersion

```
getVersion(): string
```

現在ロードされているWijmoライブラリのバージョンを取得します。

戻り値 **string**

hasClass

```
hasClass(e: Element, className: string): boolean
```

要素がクラスを持つかどうかをチェックします。

パラメーター

- **e**: **Element**
  チェックする要素。

- **className**: **string**
  チェックするクラス。

戻り値 **boolean**

hasItems

```
hasItems(value: ICollectionView): boolean
```

ICollectionView が定義されていて空でないかどうかをチェックします。

パラメーター

- **value**: **ICollectionView**
  チェックする**ICollectionView**。

戻り値 **boolean**

hidePopup

```
hidePopup(popup: HTMLElement, remove?: any, fadeOut?: boolean): any
```

showPopup メソッドによって表示されたポップアップ要素を非表示にします。

パラメーター

- **popup**: **HTMLElement**
  非表示にするポップアップ要素。

- **remove**: **any** OPTIONAL
  DOMからポップアップを削除するか、またはポップアップを非表示にするかを決定します。 このパラメータは、ポップアップがDOMから削除された後で呼び出されるコールバック関数またはブール値です。

- **fadeOut**: **boolean** OPTIONAL
  ポップアップを徐々に非表示にするフェードアウトアニメーションを使用するかどうか。

戻り値 **any**

httpRequest

```
httpRequest(url: string, settings?: any): XMLHttpRequest
```

HTTP要求を実行します。

settings パラメータ には、次の要素を含めることができます。

です。 型の1つのパラメータが渡されます。 型の1つのパラメータが渡されます。 型の1つのパラメータが渡されます。 型の1つのパラメータが渡されます。

method	.要求に使用するHTTPメソッド（"POST"、"GET"、"PUT"など）。 デフォルトは"GET".
data	サーバーに送信されるデータ。これは、GET要求の場合はurlに付加され、他の要求の場合は 文字列に変換されます。
async	デフォルトでは、すべての要求が非同期に送信されます（デフォルトではtrueに設定されます）。 同期要求が必要な場合は、このオプションをfalseに設定します。
success	要求が成功した場合に呼び出される関数。 この関数には、**XMLHttpRequest**.
error	要求が失敗した場合に呼び出される関数。 この関数には、**XMLHttpRequest**.
complete	要求の終了時（successおよびerrorコールバックが実行された後）に呼び出される関数。 この関数には、**XMLHttpRequest**.
beforeSend	要求が送信される直前に呼び出される関数。 この関数には、**XMLHttpRequest**.
requestHeaders	要求ヘッダーに追加されるキー/値のペアを含むJavaScriptオブジェクト。
user	HTTPアクセス認証要求に応答して **XMLHttpRequest** で使用されるユーザー名。
password	HTTPアクセス認証要求に応答して **XMLHttpRequest** で使用されるパスワード。

success を使用して、このコールバックの **XMLHttpRequest** パラメータで 提供される要求の結果を取得します。たとえば、次のコードは、**httpRequest** メソッドを使用して、ODataサービスから顧客のリストを取得します。

```
wijmo.httpRequest('http://services.odata.org/Northwind/Northwind.svc/Customers?$format=json', {
  success: function (xhr) {
    var response = JSON.parse(xhr.responseText),
        customers = response.value;

    // 顧客に対して何らかの処理を行います。...
  }
});
```

パラメーター

- **url**: **string**
  要求が送信されるURLを含む文字列です。

- **settings**: **any** OPTIONAL
  リクエストの構築に使用する任意指定のオブジェクト。

戻り値 **XMLHttpRequest**

isArray

```
isArray(value: any): boolean
```

オブジェクトが配列かどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

isBoolean

```
isBoolean(value: any): boolean
```

オブジェクトがブール値かどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

isDate

```
isDate(value: any): boolean
```

オブジェクトが日付かどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

isEmpty

```
isEmpty(obj: any): boolean
```

オブジェクトが空（列挙可能なプロパティが含まれない）であるかどうかを判定します。

パラメーター

- **obj**: **any**
  テストするオブジェクト。

戻り値 **boolean**

isFunction

```
isFunction(value: any): boolean
```

オブジェクトが関数かどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

isInt

```
isInt(value: any): boolean
```

オブジェクトが整数かどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

isNullOrWhiteSpace

```
isNullOrWhiteSpace(value: string): boolean
```

文字列がnull、空、またはホワイトスペースのみかどうかを判断します。

パラメーター

- **value**: **string**
  テストする値。

戻り値 **boolean**

isNumber

```
isNumber(value: any): boolean
```

オブジェクトが数値かどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

isObject

```
isObject(value: any): boolean
```

オブジェクトが（値型または日付ではなく）オブジェクトであるかどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

isPrimitive

```
isPrimitive(value: any): boolean
```

オブジェクトがプリミティブ型（string、number、Boolean、Date）かどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

isString

```
isString(value: any): boolean
```

オブジェクトが文字列かどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

isUndefined

```
isUndefined(value: any): boolean
```

オブジェクトが未定義かどうかを判断します。

パラメーター

- **value**: **any**
  テストする値。

戻り値 **boolean**

mouseToPage

```
mouseToPage(e: any): Point
```

マウスまたはタッチイベント引数をページ座標内の**`Point`** に変換します。

パラメーター

- **e**: **any**

戻り値 **`Point`**

moveFocus

```
moveFocus(parent: HTMLElement, offset: number): void
```

指定された親要素内の次/前/最初のフォーカス可能な子にフォーカスを移動します。

パラメーター

- **parent**: **HTMLElement**
  親要素。

- **offset**: **number**
  フォーカスを移動するときに使用するオフセット（フォーカス可能な最初の子要素にフォーカスする場合は、0を使用）。

戻り値 **void**

removeChild

```
removeChild(e: Node): void
```

要素をDOMツリーから安全に削除します。

パラメーター

- **e**: **Node**
  DOMツリーから削除する要素。

戻り値 **void**

removeClass

```
removeClass(e: Element, className: string): void
```

要素からクラスを削除します。

パラメーター

- **e**: **Element**
  クラスを削除する要素。

- **className**: **string**
  要素から削除するクラス（またはスペースで区切られたクラスのリスト）。

戻り値 **void**

setAriaLabel

```
setAriaLabel(e: Element, value?: string): void
```

要素の **aria-label** 属性を設定またはクリアします。

パラメーター

- **e**: **Element**
  更新する要素。
- **value**: **string** OPTIONAL
  aria labelの値。要素からラベルを削除するにはnull。

戻り値 **void**

setAttribute

```
setAttribute(e: Element, name: string, value?: any, keep?: boolean): void
```

要素の属性を設定またはクリアします。

パラメーター

- **e**: **Element**
  更新する要素。
- **name**: **string**
  追加または削除する属性の名前。
- **value**: **any** OPTIONAL
  属性の値。要素から属性を削除するにはnull。
- **keep**: **boolean** OPTIONAL
  存在する場合、元の属性を保持するかどうか。

戻り値 **void**

setCss

```
setCss(e: any, css: any): void
```

オブジェクトで指定されたプロパティを適用して、要素のスタイルを変更します。

パラメーター

- **e**: **any**
  スタイルを変更する要素または要素の配列。
- **css**: **any**
  要素に適用するスタイルプロパティを含むオブジェクト。

戻り値 **void**

setLicenseKey

```
setLicenseKey(licenseKey: string): void
```

ライセンス付きWijmoアプリケーションを識別するライセンスキーを設定します。

ライセンスキーが設定されていない場合は、Wijmoは評価モードで動作し、ページに透かしが追加されます。

ライセンスを取得済みユーザーは、Wijmoサイトの**新規ユーザー登録／ログイン**のセクションからキーを入手することができます。

Wijmoはキーやライセンス情報をどのサーバーにも送信しないことに注意してください。 これは入力されたキーの内部一貫性のみを確認します。

パラメーター

- **licenseKey**: **string**
  このアプリケーションで使用するライセンスキーを含む文字列。

戻り値 **void**

setSelectionRange

```
setSelectionRange(e: any, start: number, end?: number): boolean
```

テキストフィールド内の選択の開始位置と終了位置を設定します。

このメソッドはHTMLInputElementオブジェクトのネイティブの**setSelectionRange** メソッドに似ていますが、例外を引き起こす可能性のある条件（要素がDOM内にない、要素が無効または非表示になっている）をチェックする点が異なります。

パラメーター

- **e**: **any**
  選択するHTMLInputElementまたはHTMLTextAreaElement。
- **start**: **number**
  選択範囲の開始のテキストフィールドへのオフセット。
- **end**: **number** OPTIONAL
  選択範囲の終了のテキストフィールドへのオフセット。

戻り値 **boolean**

setText

```
setText(e: HTMLElement, text: string): void
```

要素のテキストの内容を設定します。

パラメーター

- **e**: **HTMLElement**
  内容が更新されている要素。
- **text**: **string**
  要素に割り当るテキスト。

戻り値 **void**

showPopup

```
showPopup(popup: HTMLElement, ref?: any, above?: boolean, fadeIn?: boolean, copyStyles?: boolean): any
```

要素をポップアップとして表示します。

ポップアップ要素はボディ要素の子になり、使用可能なスペースに応じて参照矩形の上または下に配置されます。

要素をポップアップとして表示します。

ポップアップ要素はボディ要素の子になり、使用可能なスペースに応じて参照矩形の上または下に配置されます。

参照矩形は以下のいずれかとして指定できます。

HTMLElement	要素の外接矩形。
MouseEvent	イベントのターゲット要素の外接矩形。
Rect	特定の矩形。
null	参照矩形なし。ポップアップはウィンドウの中央に配置されます。.

hidePopup メソッドを呼び出してポップアップを非表示にします。

パラメーター

- **popup**: **HTMLElement**
  ポップアップとして表示する要素。
- **ref**: **any** OPTIONAL
  ポップアップの配置に使用される参照要素または矩形。
- **above**: **boolean** OPTIONAL
  可能な場合、ポップアップを参照矩形の上に配置します。
- **fadeIn**: **boolean** OPTIONAL
  フェードインアニメーションを使用してポップアップを徐々に表示します。
- **copyStyles**: **boolean** OPTIONAL
  参照要素からフォントと色のスタイルをコピーします。

戻り値 **any**

toFixed

```
toFixed(value: number, prec: number, truncate: boolean): number
```

数値を指定した精度に切り上げまたは切り捨てます。

パラメーター

- **value**: **number**
  切り上げまたは切り捨てる値。
- **prec**: **number**
  結果の桁数。
- **truncate**: **boolean**
  元の値を切り捨てるか、切り上げるか。

戻り値 **number**

toggleClass

```
toggleClass(e: Element, className: string, addOrRemove?: boolean): void
```

要素にクラスを追加、または要素からクラスを削除します。

パラメーター

- **e**: **Element**
  クラスを追加する要素。
- **className**: **string**
  追加または削除するクラス。
- **addOrRemove**: **boolean** OPTIONAL
  クラスを追加するか削除するか。 指定しない場合は、クラスを切り替えます。

戻り値 **void**

toHeaderCase

```
toHeaderCase(text: string): string
```

キャメルケースの文字列をヘッダタイプの文字列に変換します。最初の文字を大文字にし、小文字の後に大文字が続いている場合に大文字の前にスペースを追加します。

たとえば、'somePropertyName'は'Some Property Name'になります。

パラメーター

- **text**: **string**
  ヘッダタイプの文字列に変換する文字列。

戻り値 **string**

tryCast

```
tryCast(value: any, type: any): any
```

可能な場合、値を型にキャストします。

パラメーター

- **value**: **any**
  キャストする値。
- **type**: **any**
  キャスト先の型またはインタフェース名。

戻り値 **any**

Binding クラス

ファイル `wijmo.js`
モジュール **wijmo**

複合プロパティ（例: 'customer.address.city'）へのバインディングを提供します。

コンストラクタ

- constructor

プロパティ

- path

メソッド

- getValue
- setValue

コンストラクタ

constructor

```
constructor(path: string): Binding
```

Binding クラスの新しいインスタンスを初期化します。

パラメーター

- **path**: **string**
  バインド先のプロパティの名前。

戻り値 **Binding**

プロパティ

- path

バインディングのパスを取得または設定します。

最も単純な場合、このパスは、連結に使用するソースオブジェクトのプロパティの 名前です（例：'street'）。

プロパティのサブプロパティは、JavaScriptと同様の構文を使用して 指定できます（例：'address.street'）。

型 **string**

メソッド

getValue

```
getValue(object: any): any
```

指定したオブジェクトのバインディング値を取得します。

オブジェクトにバインディングの**path** で指定されたプロパティが含まれていない場合、メソッドはnullを返します。

パラメーター

- **object**: **any**
  データを取得するオブジェクト。

戻り値 **any**

setValue

```
setValue(object: any, value: any): void
```

指定したオブジェクトにバインディング値を設定します。

オブジェクトにバインディングの**path** で指定されたプロパティが含まれていない場合、値は設定されません。

パラメーター

- **object**: **any**
  データを設定するオブジェクト。

- **value**: **any**
  設定するデータ値。

戻り値 **void**

CancelEventArgs クラス

ファイル wijmo.js
モジュール **wijmo**
基本クラス **EventArgs**
派生クラス **RequestErrorEventArgs**, **TooltipEventArgs**, **PageChangingEventArgs**, **CellRangeEventArgs**, **RenderEventArgs**, **TreeNodeDragDropEventArgs**, **TreeNodeEventArgs**
表示 ☐ 継承されたメンバー ☐ イベント発生元

キャンセル可能なイベントの引数を提供します。

プロパティ

- cancel
- empty

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

型 **boolean**

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

Clipboard クラス

ファイル wijmo.js
モジュール **wijmo**

クリップボード操作用のユーティリティメソッドを提供する静的クラス。

Clipboard クラスには、クリップボード操作中にクリップボードの内容をカスタマイズするためにコントロールによって使用できる静的な **copy** メソッドと**paste** メソッドがあります。

たとえば、以下のコードは、コントロールで押されたクリップボードショートカットキーをインターセプトしてカスタムのクリップボード処理を提供する方法を示します。

```
rootElement.addEventListener('keydown', function(e) {

  // コピー: [Ctrl] + [C] または [Ctrl] + [Insert]
  if (e.ctrlKey && (e.keyCode == 67 || e.keyCode == 45)) {
    var text = this.getClipString();
    Clipboard.copy(text);
    return;
  }

  // 貼り付け: [Ctrl] + [V] または [Shift] + [Insert]
  if ((e.ctrlKey && e.keyCode == 86) || (e.shiftKey && e.keyCode == 45)) {
    Clipboard.paste(function (text) {
      this.setClipString(text);
    });
    return;
  }
});
```

次の例では、ターゲットが**FlexGrid** コントロールのときにクリップボードの貼り付けコマンドの動作をカスタマイズする方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/64vr06dd)

メソッド

- copy
- paste

メソッド

STATIC copy

```
copy(text: string): void
```

文字列をクリップボードにコピーします。

このメソッドは、ユーザーがクリップボードのコピーコマンド（［Ctrl］+［C］など）を押した直後に呼び出した場合にのみ機能します。

パラメーター

- **text**: **string**
  クリップボードにコピーするテキスト。

戻り値 **void**

STATIC paste

```
paste(callback: Function): void
```

クリップボードから文字列を取得します。

このメソッドは、ユーザーがクリップボードの貼り付けコマンド（［Ctrl］+［V］など）を押した直後に呼び出した場合にのみ機能します。

パラメーター

- **callback**: **Function**
  クリップボードの内容が取得されたときに呼び出される関数。 この関数は、クリップボードの内容をパラメーターとして受け取ります。

戻り値 **void**

Color クラス

ファイル `wijmo.js`
モジュール **wijmo**

色を表します。

Color クラスは、CSS文字列として指定された色を解析し、その赤、緑、青、 およびアルファチャンネルを読み取り/書き込み可能なプロパティとして公開します。

また、**Color** クラスは、RGBではなくHSBカラーモデルとHSLカラーモデルを使用して 色を作成するための**fromHsb** メソッドと**fromHsl** メソッド、 およびそれらのカラーモデルを使用して色要素を取得するための **getHsb** メソッドと**getHsl** メソッドを提供します。

最後に、**Color** クラスは、HSLモデルを使用して2色間を補間することで 色を作成する**interpolate** メソッドを提供します。 このメソッドは、**animate** メソッドでカラーアニメーションを作成する場合に 特に便利です。

以下の例はこの仕組みを示します。

サンプル

Show me (https://jsfiddle.net/Wijmo5/xjo09z48)

コンストラクタ

- constructor

プロパティ

- a
- b
- g
- r

メソッド

- equals
- fromHsb
- fromHsl
- fromRgba
- fromString
- getHsb
- getHsl
- interpolate
- toOpaque
- toString

コンストラクタ

constructor

```
constructor(color: string): Color
```

CSS色指定から新しい**Color** を初期化します。

パラメーター

- **color**: **string**
  CSS色指定。

戻り値 **Color**

プロパティ

a

この**Color** のアルファ成分を0~1の範囲（0は透明、1は不透明）で取得または設定します。

型 **number**

b

この**Color** の青成分を0~255の範囲で取得または設定します。

型 **number**

g

この**Color** の緑成分を0～255の範囲で取得または設定します。

型 **number**

r

この**Color** の赤成分を0~255の範囲で取得または設定します。

型 **number**

メソッド

equals

```
equals(clr: Color): boolean
```

Color がこの**Color** と同じ値を持つ場合、trueを返します。

パラメーター

- **clr**: **Color**
  この**Color** と比較する**Color** 。

戻り値 **boolean**

STATIC fromHsb

```
fromHsb(h: number, s: number, b: number, a?: number): Color
```

指定したHSB値を使用して新しい**Color** を作成します。

パラメーター

- **h**: **number**
  色相（0〜1）。
- **s**: **number**
  彩度（0〜1）。
- **b**: **number**
  明度（0〜1）。
- **a**: **number** OPTIONAL
  アルファ（0〜1）。

戻り値 **Color**

STATIC fromHsl

```
fromHsl(h: number, s: number, l: number, a?: number): Color
```

指定したHSL値を使用して新しい**Color** を作成します。

パラメーター

- **h**: **number**
  色相（0〜1）。
- **s**: **number**
  彩度（0〜1）。
- **l**: **number**
  輝度（0〜1）。
- **a**: **number** OPTIONAL
  アルファ（0〜1）。

戻り値 **Color**

STATIC fromRgba

```
fromRgba(r: number, g: number, b: number, a?: number): Color
```

指定したRGBA色チャネル値を使用して新しい**`Color`** を作成します。

パラメーター

- **r**: **number**
  赤チャネルの値（0~255）。
- **g**: **number**
  緑チャネルの値（0~255）。
- **b**: **number**
  青チャネルの値（0~255）。
- **a**: **number** OPTIONAL
  アルファチャネルの値（0~1）。

戻り値 **`Color`**

STATIC fromString

```
fromString(value: string): Color
```

CSS色文字列から新しい**`Color`** を作成します。

パラメーター

- **value**: **string**
  CSS色指定を含む文字列。

戻り値 **`Color`**

getHsb

```
getHsb(): number[]
```

この色のHSB成分を含む配列を取得します。

戻り値 **number[]**

getHsl

```
getHsl(): number[]
```

この色のHSL成分を含む配列を取得します。

戻り値 **number[]**

STATIC interpolate

```
interpolate(c1: Color, c2: Color, pct: number): Color
```

2つの色を補間することによって新しい**Color** を作成します。

パラメーター

- **c1**: **Color**
  最初の色。
- **c2**: **Color**
  2番目の色。
- **pct**: **number**
  補間色を最初の色にどれだけ近づけるかを指定する0~1の値。

戻り値 **Color**

STATIC toOpaque

```
toOpaque(c: any, bkg?: any): Color
```

指定された色に最も近い不透過色を取得します。

パラメーター

- **c**: **any**
  不透過色に変換される**Color** （この色を文字列として指定することもできます）。
- **bkg**: **any** OPTIONAL
  透過色を削除したときに使用する背景色（デフォルトは白）。

戻り値 **Color**

toString

```
toString(): string
```

この**Color** の文字列表現を取得します。

戻り値 **string**

Control クラス

ファイル	`wijmo.js`
モジュール	**wijmo**
派生クラス	**FlexGrid**, **ColumnFilterEditor**, **ConditionFilterEditor**, **ValueFilterEditor**, **GroupPanel**, **Calendar**, **ColorPicker**, **DropDown**, **InputMask**, **InputNumber**, **ListBox**, **Popup**, **FlexChartBase**, **Gauge**, **TabPanel**, **TreeView**, **PivotChart**, **PivotFieldEditor**, **PivotFilterEditor**, **PivotPanel**, **Slicer**, **ViewerBase**
表示	☐継承されたメンバー ☐イベント発生元

すべてのWijmoコントロールの基本クラス。

Control クラスは、DOM要素と実際のコントロールとの関連を処理します。コントロールをホストしているDOM要素を取得するには **hostElement** プロパティを使用し、 特定のDOM要素でホストされているコントロールを取得するには**getControl** メソッドを使用します。

さらに、**Control** クラスは、コントロールの無効化と更新、コントロールのサイズが変更されたときのコントロールレイアウトの更新、 およびコントロールの構造を定義するHTMLテンプレートの処理に関する共通のパターンを提供します。

コンストラクタ

- constructor

プロパティ

- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, options?, invalidateOnResize?: boolean): Control
```

Control クラスの新しいインスタンスを初期化してDOM要素にアタッチします。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。
- **invalidateOnResize**: **boolean** OPTIONAL
  コントロールのサイズが変更されたときにコントロールを無効にするかどうか。

戻り値 **Control**

プロパティ

hostElement

コントロールをホストしているDOM要素を取得します。

型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**`Control`** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

引数 **EventArgs**

DateTime クラス

ファイル wijmo.js
モジュール **wijmo**

日付および時刻のユーティリティを提供します。

メソッド

- addDays
- addHours
- addMinutes
- addMonths
- addSeconds
- addYears
- clone
- equals
- fromDateTime
- fromFiscal
- newDate
- sameDate
- sameTime
- toFiscal

メソッド

- STATIC addDays

```
addDays(value: Date, days: number): Date
```

特定の日付に指定した日数を加算した新しい日付を取得します。

パラメーター

- **value**: **Date**
  元の日付。
- **days**: **number**
  指定した日付に加算する日数。

戻り値 **Date**

STATIC addHours

```
addHours(value: Date, hours: number): Date
```

特定の日付に指定した時間数を加算した新しい日付を取得します。

パラメーター

- **value**: **Date**
  元の日付。
- **hours**: **number**
  指定した日付に加算する時間数。

戻り値 **Date**

STATIC addMinutes

```
addMinutes(value: Date, minutes: number): Date
```

特定の日付に指定した分数を加算した新しい日付を取得します。

パラメーター

- **value**: **Date**
  元の日付。
- **minutes**: **number**
  指定した日付に加算する分数。

戻り値 **Date**

STATIC addMonths

```
addMonths(value: Date, months: number): Date
```

特定の日付に指定した月数を加算した新しい日付を取得します。

パラメーター

- **value**: **Date**
  元の日付。
- **months**: **number**
  指定した日付に加算する月数。

戻り値 **Date**

STATIC addSeconds

```
addSeconds(value: Date, seconds: number): Date
```

特定の日付に指定した秒数を加算した新しい日付を取得します。

パラメーター

- **value**: **Date**
  元の日付。

- **seconds**: **number**
  指定した日付に加算する秒数。

戻り値 **Date**

STATIC addYears

```
addYears(value: Date, years: number): Date
```

特定の日付に指定した年数を加算した新しい日付を取得します。

パラメーター

- **value**: **Date**
  元の日付。

- **years**: **number**
  指定した日付に加算する年数。

戻り値 **Date**

STATIC clone

```
clone(date: Date): Date
```

指定したDate オブジェクトのコピーを作成します。

パラメーター

- **date**: **Date**
  コピーするDateオブジェクト。

戻り値 **Date**

STATIC equals

```
equals(d1: Date, d2: Date): boolean
```

2つのDateオブジェクトが同じ日付と時刻を指している場合、trueを返します。

パラメーター

- **d1**: **Date**
  最初の日付。

- **d2**: **Date**
  2番目の日付。

戻り値 **boolean**

STATIC fromDateTime

```
fromDateTime(date: Date, time: Date): Date
```

2つのDateオブジェクトに設定された日付と時刻を持つDateオブジェクトを取得します。

パラメーター

- **date**: **Date**
  日付（年/月/日）を含むDateオブジェクト。

- **time**: **Date**
  時刻（時:分:秒）を含むDateオブジェクト。

戻り値 **Date**

STATIC fromFiscal

```
fromFiscal(date: Date, govt: boolean): void
```

現在のカルチャを使用して、会計年度日付をカレンダー日付に変換します。

パラメーター

- **date**: **Date**
  会計年度の日付。

- **govt**: **boolean**
  政府の会計年度と企業の会計年度のどちらを使用するか。

戻り値 **void**

STATIC newDate

```
newDate(year?: number, month?: number, day?: number, hour?: number, min?: number, sec?: number, ms?: number): Date
```

新しいDateオブジェクトインスタンスを取得します。

パラメーター

- **year**: **number** OPTIONAL
  年を表す整数値。デフォルトの値は現在の年です。

- **month**: **number** OPTIONAL
  月(0-11)を表す整数値。デフォルトの値は現在の月です。

- **day**: **number** OPTIONAL
  月の日付(1-31)を表す整数値。デフォルトの値は現在の年です。

- **hour**: **number** OPTIONAL
  時間を表す整数値。デフォルトの値は0です。

- **min**: **number** OPTIONAL
  分を表す整数値。デフォルトの値は0です。

- **sec**: **number** OPTIONAL
  秒を表す整数値。デフォルトの値は0です。

- **ms**: **number** OPTIONAL
  ミリ秒を表す整数値。デフォルトの値は0です。

戻り値 **Date**

STATIC sameDate

```
sameDate(d1: Date, d2: Date): boolean
```

2つのDateオブジェクトが同じ日付を指している場合、trueを返します（時刻は無視します）。

パラメーター

- **d1**: **Date**
  最初の日付。
- **d2**: **Date**
  2番目の日付。

戻り値 **boolean**

STATIC sameTime

```
sameTime(d1: Date, d2: Date): boolean
```

2つのDateオブジェクトが同じ時刻を指している場合、trueを返します（日付は無視します）。

パラメーター

- **d1**: **Date**
  最初の日付。
- **d2**: **Date**
  2番目の日付。

戻り値 **boolean**

STATIC toFiscal

```
toFiscal(date: Date, govt: boolean): void
```

現在のカルチャを使用して、カレンダー日付を会計日付に変換します。

パラメーター

- **date**: **Date**
  カレンダーの日付。
- **govt**: **boolean**
  政府の会計年度と企業の会計年度のどちらを使用するか。

戻り値 **void**

Event クラス

ファイル wijmo.js
モジュール **wijmo**

イベントを表します。

Wijmoのイベントは.NETのイベントに似ています。どのクラスでも、フィールドとして宣言することによってイベントを定義できます。クラスでイベントをサブスクライブするには、イベントの**addHandler** メソッドを使用し、 アンサブスクライブするには、**removeHandler** メソッドを使用します。

Wijmoのイベントハンドラは、 *sender*と*args*の2つのパラメーターをとります。 最初のパラメータは、イベントを発生させたオブジェクトです。２番目のパラメータは、イベントパラメータを含むオブジェクトです。

イベントを定義するクラスは、イベントごとにイベントを発生させる *on[EVENTNAME]*メソッドがあるという.NETのパターンに従います。このパターンに従うと、派生クラスで*on[EVENTNAME]*メソッドをオーバーライドして、基本クラスがそのイベントを発生させる前または後にイベントを処理できます。派生クラスでは、基本クラスの実装を呼び出さないようにすることで、イベントを抑止することもできます。

たとえば、次のTypeScriptコードは、コントロールの**onValueChanged**イベントをオーバーライドして、 **valueChanged**イベントが発生する前後に何らかの処理を実行します。

```
// 基本クラスをオーバーライドします
onValueChanged(e: EventArgs) {

  // イベントが発生する前に何らかのコードを実行します
  console.log('about to fire valueChanged');

  // オプションで、基本クラスを呼び出してイベントを発生させます
  super.onValueChanged(e);

  // イベントが発生した後に何らかのコードを実行します
  console.log('valueChanged event just fired');
}
```

プロパティ

- handlerCount
- hasHandlers

メソッド

- addHandler
- raise
- removeAllHandlers
- removeHandler

プロパティ

- handlerCount

このイベントに追加されたハンドラの数を取得します。

型 **number**

- hasHandlers

このイベントがハンドラを持つかどうかを示す値を取得します。

型 **boolean**

メソッド

addHandler

```
addHandler(handler: IEventHandler, self?: any): void
```

このイベントにハンドラを追加します。

パラメーター

- **handler**: **IEventHandler**
  イベントが発生したときに呼び出される関数。

- **self**: **any** OPTIONAL
  イベントハンドラを定義するオブジェクト（ハンドラのコードから'this'としてアクセス可能）。

戻り値 **void**

raise

```
raise(sender: any, args?: EventArgs): void
```

このイベントを発生させます。これにより、関連付けられたすべてのハンドラが呼び出されます。

パラメーター

- **sender**: **any**
  ソースオブジェクト。

- **args**: **EventArgs** OPTIONAL
  イベントパラメーター。

戻り値 **void**

removeAllHandlers

```
removeAllHandlers(): void
```

このイベントに関連付けられたすべてのハンドラを削除します。

戻り値 **void**

removeHandler

```
removeHandler(handler: IEventHandler, self?: any): void
```

このイベントからハンドラを削除します。

パラメーター

- **handler**: **IEventHandler**
  イベントが発生したときに呼び出される関数。

- **self**: **any** OPTIONAL
  イベントハンドラを定義するオブジェクト（ハンドラのコードから'this'としてアクセス可能）。

戻り値 **void**

EventArgs クラス

ファイル wijmo.js
モジュール **wijmo**
派生クラス **CancelEventArgs**, **PropertyChangedEventArgs**, **NotifyCollectionChangedEventArgs**, **FormatItemEventArgs**, **SeriesEventArgs**, **PdfDocumentEndedEventArgs**, **PdfWebWorkerExportDoneEventArgs**, **FormatNodeEventArgs**, **DraggingRowColumnEventArgs**, **RowColumnChangedEventArgs**, **UnknownFunctionEventArgs**, **ProgressEventArgs**, **QueryLoadingDataEventArgs**, **RequestEventArgs**

イベント引数の基本クラス。

プロパティ

- empty

プロパティ

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

型 **EventArgs**

Globalize クラス

ファイル wijmo.js
モジュール **wijmo**

数値および日付の書式設定と解析を実装するクラス。

デフォルトでは、**Globalize** はアメリカ英語カルチャを使用します。 カルチャを切り替えるには、該当する**wijmo.culture.*.js**ファイルをwijmoファイルの後にインクルードします。

次の例では、**Globalize** クラスを使用して、さまざまなカルチャの日付、時刻および数値の書式を設定する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/u9fo3ynp)

メソッド

- format
- formatDate
- formatNumber
- getFirstDayOfWeek
- getNumberDecimalSeparator
- parseDate
- parseFloat
- parseInt

メソッド

STATIC format

```
format(value: any, format: string, trim?: boolean, truncate?: boolean): string
```

数値または日付を書式設定します。

format 関数で使用される書式文字列は、**Globalize.js**および.NETグローバリゼーションライブラリで使用される書式とほぼ同じです。使用可能な書式について説明するリンクを以下に示します。

- 標準の数値書式指定文字列 (http://msdn.microsoft.com/ja-jp/library/dwhawy9k(v=vs.110).aspx)
- 標準の日時書式指定文字列 (http://msdn.microsoft.com/ja-jp/library/az4se3k1(v=vs.110).aspx)
- カスタムの日時書式指定文字列 (http://msdn.microsoft.com/ja-jp/library/8kb3ddd4(v=vs.110).aspx)

パラメーター

- **value**: **any**
  書式設定する数値または日付（その他すべての型は文字列に変換されます）。
- **format**: **string**
  数値または日付の書式設定に使用する書式文字列。
- **trim**: **boolean** OPTIONAL
  数値の結果から後続のゼロを除去するかどうか。
- **truncate**: **boolean** OPTIONAL
  数値を切り捨てるか、切り上げるか。

戻り値 **string**

STATIC formatDate

```
formatDate(value: Date, format: string): string
```

現在のカルチャを使用して日付を書式設定します。

format パラメータには、.NET形式の **日付書式文字列** と次の追加要素が含まれます。

- *Q, q* カレンダー四半期。
- *U* 会計四半期（政府）。
- *u* 会計四半期（民間）。
- *EEEE, EEE, EE, E* 会計年度（政府）。
- *eeee, eee, ee, e* 会計年度（民間）。

次に例を示します。

```
var d = new Date(2015, 9, 1); // 2015年10月1日
console.log(wijmo.Globalize.format(d, '"FY"EEEE"Q"U') + ' (US culture)');
> FY2016Q1 (US culture)
```

Another addition is available for dealing with complex eras such as those defined in the Japanese culture:

- *ggg* Era name (e.g. '平成', '昭和', '大正', or '明治').
- *gg* Era initial (e.g. '平', '昭', '大', or '明').
- *g* Era symbol (e.g. 'H', 'S', 'T', or 'M').

サンプル

Show me (https://jsfiddle.net/Wijmo5/vw6en3sa)

パラメーター

- **value**: **Date**
  書式設定する数値または日付。
- **format**: **string**
  .NET形式の日付書式文字列。

戻り値 **string**

STATIC formatNumber

```
formatNumber(value: number, format: string, trim?: boolean, truncate?: boolean): string
```

現在のカルチャを使用して数値を書式設定します。

formatNumber メソッドは、ほとんどの.NET形式の **標準数値書式文字列 (http://msdn.microsoft.com/ja-JP/library/dwhawy9k(v=vs.110).aspx)**を受け入れます。ただし、'e'および'x'書式 （指数表記および16進数）はサポートされていません。

数値書式文字列の形式は *Axxsscc*です。

- *A*は、1つのアルファベット文字で表される書式指定子です。 大文字小文字は区別されません。
- *xx*は、整数で表されるオプションの精度指定子です。 精度指定子は、結果の桁数に影響します。
- *ss*は、数値のスケールするためのオプションの文字列です。これを指定する場合は、カンマで構成する必要があります。数値が指定されたカンマごとに1000で除算されます。.
- *cc*は、通貨値の書式設定を行う際に、通貨シンボルをオーバーライドするために使用されるオプションの文字列です。これは、現在のデフォルトとは異なる カルチャの通貨値を書式設定する場合に役立ちます （たとえば、英語カルチャを使用するアプリケーションでユーロや円の値を書式設定する場合）

次の表では、標準の数値書式指定子について説明し、 またデフォルトのカルチャの各書式指定子によって 生成される出力例を表示します。

n 数値： `formatNumber(1234.5, 'n2') => '1,234.50'`
f 固定小数点： `formatNumber(1234.5, 'f2') => '1234.50'`
g 汎用（末尾のゼロなし）： `formatNumber(1234.5, 'g2') => '1234.5'`
d 10進数（整数）： `formatNumber(-1234, 'd6') => '-001234'`
x 16進数（整数）： `formatNumber(1234, 'x6') => '0004d2'`
c 通貨： `formatNumber(1234, 'c') => '$ 1,234.00'`
p パーセント： `formatNumber(0.1234, 'p2') => '12.34 %'`

単位変更指定子は、大きな値をチャートに表示する場合に特に便利です。たとえば、次のマークアップは、人口とGDPをプロットするチャートを作成します。 生のデータでは、人口は人数そのもの、GDPは100万単位です。 軸の書式設定で指定された単位変更に基づいて、人口は100万単位、GDPは兆単位で表示されます。

```
<wj-flex-chart
  items-source="countriesGDP" binding-x="pop" chart-type="Scatter">
  <wj-flex-chart-series
    name="GDP" binding="gdp"></wj-flex-chart-series>
  <wj-flex-chart-axis
    wj-property="axisX" title="Population (millions)"
    format="n0,,">
  </wj-flex-chart-axis>
  <wj-flex-chart-axis
    wj-property="axisY" title="GDP (US$ trillions)"
    format="c0,,">
  </wj-flex-chart-axis>
</wj-flex-chart>
```

パラメーター

- **value**: **number**
  書式設定する数値。

- **format**: **string**
  .NETスタイルの標準の数値書式文字列（例: 'n2'、'c4'、'p0'、'g2'、'd2'）。

- **trim**: **boolean** OPTIONAL
  結果から後続のゼロを除去するかどうか。

- **truncate**: **boolean** OPTIONAL
  値を切り捨てるか、切り上げるか。

戻り値 **string**

STATIC getFirstDayOfWeek

```
getFirstDayOfWeek(): number
```

現在のカルチャに従って週の最初の曜日を取得します。

返される値は0（日曜日）~6（土曜日）です。

戻り値 **number**

STATIC getNumberDecimalSeparator

```
getNumberDecimalSeparator(): string
```

数値の小数点記号として使用される記号を取得します。

戻り値 **string**

STATIC parseDate

```
parseDate(value: string, format: string, defaultDate?: Date): Date
```

文字列を日付に解析します。

2桁の年は、カレンダーの**twoDigitYearMax**プロパティの値に基づいて、 完全な年に変換されます。デフォルトでは、このプロパティは2029に設定されています。 つまり、2桁の30～99の値は19**になり、 0～29の値は20**になります。

このしきい値は、カレンダーに新しい値を割り当てて変更できます。 次に例を示します。

```
// カレンダーを取得します
var cal = wijmo.culture.Globalize.calendar;

// デフォルトのしきい値は2029なので、"30"は1930と解析されます
cal.twoDigitYearMax = 2029;
var d1 = wijmo.Globalize.parseDate('30/12', 'yy/MM'); // dec 1930
// しきい値を2100に変更すると、すべての値が20**と解析されます
cal.twoDigitYearMax = 2100;
var d2 = wijmo.Globalize.parseDate('30/12', 'yy/MM'); // dec 2030
```

パラメーター

- **value**: **string**
  日付に変換する文字列。
- **format**: **string**
  日付の解析に使用する書式文字列。
- **defaultDate**: **Date** OPTIONAL
  日付部分が入力内容から欠けている場合(たとえば、format = 'MM/dd')に参照として使用する日付。

戻り値 **Date**

STATIC parseFloat

```
parseFloat(value: string, format?: string): number
```

文字列を浮動小数点数に解析します。

パラメーター

- **value**: **string**
  数値に変換する文字列。

- **format**: **string** OPTIONAL
  数値の解析時に使用する書式。

戻り値 **number**

STATIC parseInt

```
parseInt(value: string, format?: string): number
```

文字列を整数に解析します。

パラメーター

- **value**: **string**
  整数に変換する文字列。

- **format**: **string** OPTIONAL
  数値の解析時に使用する書式。

戻り値 **number**

Point クラス

ファイル wijmo.js
モジュール **wijmo**

x座標とy座標を持つ点を表すクラス。

コンストラクタ

- constructor

プロパティ

- x
- y

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(x?: number, y?: number): Point
```

Point クラスの新しいインスタンスを初期化します。

パラメーター

- **x**: **number** OPTIONAL
  新しいPointのX座標。
- **y**: **number** OPTIONAL
  新しいPointのY座標。

戻り値 **Point**

プロパティ

- x

このPoint のx座標を取得または設定します。

型 **number**

y

このPoint のy座標を取得または設定します。

型 **number**

メソッド

clone

```
clone(): Point
```

この**Point** のコピーを作成します。

戻り値 **Point**

equals

```
equals(pt: Point): boolean
```

Point がこの**Point** と同じ座標を持つ場合、trueを返します。

パラメーター

- **pt**: **Point**
  この**Point** と比較する**Point** 。

戻り値 **boolean**

PrintDocument クラス

ファイル wijmo.js
モジュール wijmo

印刷用のカスタムドキュメントの作成を可能にするクラス。

PrintDocument クラスを使用すると、印刷用またはPDFへのエクスポート用の ドキュメントを簡単に作成できます。大部分のブラウザでは、用紙サイズ、方向、マージン、および ページヘッダー/フッターを含むかどうかを選択できます。

使用するには、**PrintDocument** をインスタンス化し、**append** メソッドを使用して コンテンツを追加し、**print** メソッドを呼び出して終了します。

次に例を示します。

```
// ドキュメントを作成します
var doc = new wijmo.PrintDocument({
  title: 'PrintDocument Test'
});

// 単純なテキストを追加します
doc.append('<h1>印刷の例</h1>');
doc.append('<p>このドキュメントは、<b>PrintDocument</b> クラスを使用して作成されました</p>');

// 既存の要素を追加します
doc.append(document.getElementById('gaugeControl'));

// ドキュメントを印刷します（またはPDFにエクスポートします）
doc.print();
```

次の例では、ブラウザからPDFやその他の形式で直接印刷またはエクスポートできる、印刷用ドキュメントを作成する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/c75xjs11)

コンストラクタ

- constructor

プロパティ

- copyCss
- title

メソッド

- addCSS
- append
- print

コンストラクタ

constructor

```
constructor(options?: any): PrintDocument
```

PrintDocument クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  PrintDocument の初期化データを含むJavaScriptオブジェクト。

戻り値 **PrintDocument**

プロパティ

copyCss

PrintDocument にメインドキュメントで定義されているCSSスタイルシートを含めるかどうかを決定する値を取得または設定します。

The default value for the property is **true**.

型 **boolean**

title

ドキュメントタイトルを取得または設定します。

このプロパティをnullに設定すると、**PrintDocument** は現在のドキュメントのタイトルを使用します。

型 **string**

メソッド

addCSS

```
addCSS(href: string): void
```

ドキュメントにCSSスタイルシートを追加します。

パラメーター

- **href**: **string**
  ドキュメントに追加するCSSファイルのURL。

戻り値 **void**

append

```
append(child: any): void
```

ドキュメントにHTML要素または文字列を追加します。

パラメーター

- **child**: **any**
  ドキュメントに付加するHTML要素または文字列。

戻り値 **void**

print

```
print(): void
```

ドキュメントを印刷します。

戻り値　　**void**

PropertyChangedEventArgs クラス

ファイル wijmo.js
モジュール wijmo
基本クラス EventArgs
表示 ☐継承されたメンバー ☐イベント発生元

プロパティ変更イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- empty
- newValue
- oldValue
- propertyName

コンストラクタ

constructor

```
constructor(propertyName: string, oldValue: any, newValue: any): PropertyChangedEventArgs
```

PropertyChangedEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **propertyName**: **string**
  値が変更されたプロパティの名前。
- **oldValue**: **any**
  プロパティの古い値。
- **newValue**: **any**
  プロパティの新しい値。

戻り値 **PropertyChangedEventArgs**

プロパティ

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

newValue

プロパティの新しい値を取得します。

型 **any**

oldValue

プロパティの古い値を取得します。

型 **any**

propertyName

値が変更されたプロパティの名前を取得します。

型 **string**

Rect クラス

ファイル wijmo.js
モジュール **wijmo**

左座標、上座標、幅、高さを持つ矩形を表すクラス。

コンストラクタ

- constructor

プロパティ

- bottom
- height
- left
- right
- top
- width

メソッド

- clone
- contains
- equals
- fromBoundingRect
- inflate
- intersection
- union

コンストラクタ

constructor

```
constructor(left: number, top: number, width: number, height: number): Rect
```

Rect クラスの新しいインスタンスを初期化します。

パラメーター

- **left**: **number**
  新しい**Rect** の左座標。
- **top**: **number**
  新しい**Rect** の上座標。
- **width**: **number**
  新しい**Rect** の幅。
- **height**: **number**
  新しい**Rect** の高さ。

戻り値 **Rect**

プロパティ

- bottom

このRect の下座標を取得します。

型 **number**

- height

このRect の高さを取得または設定します。

型 **number**

- left

このRect の左座標を取得または設定します。

型 **number**

- right

このRect の右座標を取得します。

型 **number**

- top

このRect の上座標を取得または設定します。

型 **number**

- width

このRect の幅を取得または設定します。

型 **number**

メソッド

- clone

```
clone(): Rect
```

このRect のコピーを作成します。

戻り値 **Rect**

contains

```
contains(pt: any): boolean
```

この矩形が指定したポイントまたは矩形を含むかどうかを調べます。

パラメーター

- **pt**: **any**
  チェックする**Point** または**Rect** 。

戻り値 **boolean**

equals

```
equals(rc: Rect): boolean
```

Rect の座標とサイズがこの**Rect** と同じ場合は、trueを返します。

パラメーター

- **rc**: **Rect**
  この**Rect** と比較する**Rect**。

戻り値 **boolean**

STATIC fromBoundingRect

```
fromBoundingRect(rc: any): Rect
```

ClientRectまたは**SVGRect**オブジェクトから**Rect** を作成します。

パラメーター

- **rc**: **any**
  DOMの**getBoundingClientRect**メソッドまたは**GetBoundingBox**メソッドの呼び出しによって取得された矩形。

戻り値 **Rect**

inflate

```
inflate(dx: number, dy: number): Rect
```

矩形を指定した量だけ拡大または縮小した結果の矩形を作成します。

パラメーター

- **dx**: **number**
  矩形を左右方向に拡大または縮小する量。

- **dy**: **number**
  矩形を上下方向に拡大または縮小する量。

戻り値 **Rect**

STATIC intersection

```
intersection(rc1: Rect, rc2: Rect): Rect
```

2つの四角形の共通部分を表す四角形を取得します。

パラメーター

- **rc1**: **Rect**
  最初の矩形。

- **rc2**: **Rect**
  2番目の矩形。

戻り値 `Rect`

STATIC union

```
union(rc1: Rect, rc2: Rect): Rect
```

2つの矩形の和集合を表す矩形を取得します。

パラメーター

- **rc1**: **Rect**
  最初の矩形。

- **rc2**: **Rect**
  2番目の矩形。

戻り値 `Rect`

RequestErrorEventArgs クラス

ファイル wijmo.js
モジュール **wijmo**
基本クラス **CancelEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

XMLHttpRequest エラーイベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- empty
- message
- request

コンストラクタ

constructor

```
constructor(xhr: XMLHttpRequest, msg?: string): RequestErrorEventArgs
```

RequestErrorEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **xhr**: **XMLHttpRequest**
  エラーを検出する**XMLHttpRequest**。要求オブジェクトのstatusとstatusTextプロパティは、エラーの詳細が含まれています。
- **msg**: **string** OPTIONAL
  オプションのエラーメッセージ。

戻り値 **RequestErrorEventArgs**

プロパティ

cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

- message

ユーザーに表示するエラーメッセージを取得または設定します。

型 **string**

- request

エラーを検出したの**XMLHttpRequest** への参照を取得します。

要求オブジェクトのstatusとstatusTextプロパティは、エラーの詳細が含まれています。

型 **XMLHttpRequest**

Size クラス

ファイル wijmo.js
モジュール **wijmo**

幅と高さを持つサイズを表すクラス。

コンストラクタ

- constructor

プロパティ

- height
- width

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(width?: number, height?: number): Size
```

Size クラスの新しいインスタンスを初期化します。

パラメーター

- **width**: **number** OPTIONAL
  新しい**Size** の幅。
- **height**: **number** OPTIONAL
  新しい**Size** の高さ。

戻り値 **Size**

プロパティ

- height

このSize の高さを取得または設定します。

型 **number**

- width

このSize の幅を取得または設定します。

型 **number**

メソッド

clone

```
clone(): Size
```

この**Size** のコピーを作成します。

戻り値 **Size**

equals

```
equals(sz: Size): boolean
```

Size がこの**Size** と同じサイズを持つ場合、trueを返します。

パラメーター

- **sz**: **Size**
  この**Size** と比較する**Size**。

戻り値 **boolean**

Tooltip クラス

ファイル wijmo.js
モジュール **wijmo**
派生クラス **ChartTooltip**

ページ上の要素に関する追加情報を表示するポップアップウィンドウを提供します。

Tooltip クラスは以下の2つのモードで使用できます。 **自動モード: setTooltip** メソッドを使用して、 **Tooltip** をページの1つ以上の要素に接続します。**Tooltip** がイベントを自動的に監視し、 ユーザーがツールチップをトリガーするアクションを実行したときにツールチップを表示します。 例:

```
var tt = new wijmo.Tooltip();
tt.setTooltip('#menu', 'Select commands.');
tt.setTooltip('#tree', 'Explore the hierarchy.');
tt.setTooltip('#chart', '#idChartTooltip');
```

手動モード: 呼び出し元が**show** メソッドと**hide** メソッドを使用してツールチップの表示/非表示を制御します。次に例を示します。

```
var tt = new wijmo.Tooltip();
element.addEventListener('click', function () {
  if (tt.isVisible) {
    tt.hide();
  } else {
    tt.show(element, 'これは必要な要素です。');
  }
});
```

次の例は、**Tooltip** クラスを使用してExcelスタイルのメモを **FlexGrid** コントロールのセルに追加する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/2d0jhd6r)

コンストラクタ

- constructor

プロパティ

- gap
- hideDelay
- isContentHtml
- isVisible
- showAtMouse
- showDelay

メソッド

- dispose
- getTooltip
- hide
- onPopup
- setTooltip
- show

イベント

- popup

コンストラクタ

constructor

```
constructor(options?: any): Tooltip
```

Tooltip クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  Tooltip の初期化データを含むJavaScriptオブジェクト。

戻り値 **Tooltip**

プロパティ

gap

ツールチップとターゲット要素との距離を取得または設定します。

The default value for the property is **6** pixels.

型 **number**

hideDelay

Gets or sets the delay, in milliseconds, before hiding the tooltip if the mouse remains over the element.

The default value for the property is **zero** milliseconds, which causes the tip to remain visible until the mouse moves away from the element.

型 **number**

isContentHtml

ツールチップのコンテンツをプレーンテキストとして表示するか、HTMLとして表示するかを決定する値を取得または設定します。

The default value for the property is **true**.

型 **boolean**

isVisible

ツールチップが表示されているかどうかを示す値を取得します。

型 **boolean**

showAtMouse

ツールチップをターゲット要素ではなくマウスの位置を基準にして配置するかどうかを決定する値を取得または設定します。

The default value for the property is **false**.

型 **boolean**

showDelay

マウスがターゲット要素に入ってからツールチップが表示されるまでの遅延（ミリ秒単位）を取得または設定します。

The default value for the property is **500** milliseconds.

型 **number**

メソッド

dispose

```
dispose(): void
```

この**Tooltip** インスタンスに関連付けられたすべてのツールチップを削除します。

戻り値 **void**

getTooltip

```
getTooltip(element: any): string
```

指定した要素に関連付けられたツールチップの内容を取得します。

パラメーター

- **element**: **any**
  ツールチップで説明する要素、要素ID、またはコントロール。

戻り値 **string**

hide

```
hide(): void
```

ツールチップが現在表示されている場合、非表示にします。

戻り値 **void**

onPopup

```
onPopup(e: TooltipEventArgs): boolean
```

popup イベントを発生させます。

パラメーター

- **e**: **TooltipEventArgs**
  イベントデータを含む**TooltipEventArgs**。

戻り値 **boolean**

setTooltip

```
setTooltip(element: any, content: string): void
```

ページ上の指定した要素にツールチップの内容を割り当てます。

ページ上の任意の数の要素に対して同じツールチップを使用して情報を表示できます。要素からツールチップを削除するには、 **setTooltip** を呼び出して内容をnullに設定します。

パラメーター

- **element**: **any**
  ツールチップで説明する要素、要素ID、またはコントロール。
- **content**: **string**
  ツールチップの内容、またはツールチップの内容を含む要素のID。

戻り値 **void**

show

```
show(element: any, content: string, bounds?: Rect): void
```

指定した要素の横に、指定した内容を含むツールチップを表示します。

パラメーター

- **element**: **any**
  ツールチップで説明する要素、要素ID、またはコントロール。
- **content**: **string**
  ツールチップの内容、またはツールチップの内容を含む要素のID。
- **bounds**: **Rect** OPTIONAL
  （オプション）ツールチップが対象とする領域の範囲を定義する要素。 これを指定しない場合は、（**getBoundingClientRect**メソッドによって報告される）要素の範囲が使用されます。

戻り値 **void**

イベント

popup

ツールチップの内容が表示される前に発生します。

イベントハンドラでイベントパラメーターを変更してツールチップの内容をカスタマイズしたり、ツールチップの表示を抑制したりできます。

引数 **TooltipEventArgs**

TooltipEventArgs クラス

ファイル wijmo.js
モジュール **wijmo**
基本クラス **CancelEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

popup イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- content
- empty

コンストラクタ

constructor

```
constructor(content: string): TooltipEventArgs
```

TooltipEventArgs クラスの新しいインスタンスを初期化します。

パラメーター
- **content**: **string**
  ツールチップに表示する文字列。

戻り値 **TooltipEventArgs**

プロパティ

cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

content

ツールチップに表示する内容を取得または設定します。

popup イベントの処理中に、このパラメータを使用して、 ツールチップのコンテンツを変更できます。

型 **string**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

IEventHandler インターフェイス

ファイル `wijmo.js`
モジュール **wijmo**

イベントハンドラを表します。

イベントハンドラは、イベントが発生したときに呼び出される関数です。

各イベントハンドラは、次の2つの引数を持ちます。

- **sender**は、イベントを発生させたオブジェクトです。
- **args**は、イベントパラメータを含むオプションのオブジェクトです。

次の例では、Wijmoのイベントにイベントハンドラを追加する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/9tkuuf5t)

IQueryInterface インターフェイス

ファイル wijmo.js
モジュール **wijmo**

オブジェクトがインタフェースを実装しているかどうかを呼び出し元が確認できるようにします。

メソッド

- implementsInterface

メソッド

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

オブジェクトが指定したインタフェースを実装している場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

戻り値 **boolean**

Aggregate 列挙体

ファイル wijmo.js
モジュール **wijmo**

値のグループに対して計算を行う集計のタイプを指定します。

メンバー

名前	値	説明
None	0	集計なし。
Sum	1	グループ内の数値の合計を返します。
Cnt	2	グループ内のnullでない値の数を返します。
Avg	3	グループ内の数値の平均値を返します。
Max	4	グループ内の最大値を返します。
Min	5	グループ内の最小値を返します。
Rng	6	グループ内の最大の数値と最小の数値の差を返します。
Std	7	グループ内の数値の標本標準偏差を返します（n-1に基づく数式を使用します）。
Var	8	グループ内の数値の標本分散を返します（n-1に基づく数式を使用します）。
StdPop	9	グループ内の数値の母標準偏差を返します（nに基づく数式を使用します）。
VarPop	10	グループ内の数値の母分散を返します（nに基づく数式を使用します）。
CntAll	11	グループ内のすべての値の数を返します（nullを含む）。
First	12	グループ内の最初のnullでない値を返します。
Last	13	グループ内の最後のnullでない値を返します。

DataType 列挙体

ファイル `wijmo.js`
モジュール **wijmo**

データ型を表す定数を指定します。

値から**DataType** を取得するには、**getType** メソッドを使用します。

メンバー

名前	値	説明
Object	0	任意のオブジェクト。
String	1	文字列。
Number	2	数値。
Boolean	3	ブール値。
Date	4	日付および時刻。
Array	5	配列。

Key 列挙体

ファイル `wijmo.js`
モジュール **wijmo**

キーボードコードを表す定数を指定します。

この列挙体は**keyDown**イベントを処理するときに役立ちます。

メンバー

名前	値	説明
Back	8	[BackSpace]キー。
Tab	9	[Tab]キー。
Enter	13	[Enter]キー。
Escape	27	[Esc]キー。
Space	32	スペースキー。
PageUp	33	[PageUp]キー。
PageDown	34	[PageDown]キー。
End	35	[End]キー。
Home	36	[Home]キー。
Left	37	[←]キー。
Up	38	[↑]キー。
Right	39	[→]キー。
Down	40	[↓]キー。
Delete	46	[Delete]キー。
F1	112	[F1]キー。
F2	113	[F2]キー。
F3	114	[F3]キー。
F4	115	[F4]キー。
F5	116	[F5]キー。
F6	117	[F6]キー。
F7	118	[F7]キー。
F8	119	[F8]キー。
F9	120	[F9]キー。
F10	121	[F10]キー。
F11	122	[F11]キー。
F12	123	[F12]キー。

wijmo.collections モジュール

ファイル wijmo.js
モジュール **wijmo.collections**

ICollectionView インタフェース、 **CollectionView** クラス、**ObservableArray** クラスなど、 データに関連するインタフェースとクラスを定義します。

クラス

- ArrayBase
- CollectionView
- CollectionViewGroup
- GroupDescription
- NotifyCollectionChangedEventArgs
- ObservableArray
- PageChangingEventArgs
- PropertyGroupDescription
- SortDescription

インターフェイス

- ICollectionView
- IComparer
- IEditableCollectionView
- INotifyCollectionChanged
- IPagedCollectionView
- IPredicate

列挙体

- NotifyCollectionChangedAction

ArrayBase クラス

ファイル wijmo.js
モジュール **wijmo.collections**
派生クラス **ObservableArray**
表示 ☐ 継承されたメンバー ☐ イベント発生元

通知機能のあるArrayクラスの基本クラス。

コンストラクタ

- constructor

コンストラクタ

constructor

```
constructor(): ArrayBase
```

ArrayBase クラスの新しいインスタンスを初期化します。

戻り値 **ArrayBase**

CollectionView クラス

ファイル	`wijmo.js`
モジュール	**wijmo.collections**
派生クラス	**ODataCollectionView**, **PivotCollectionView**
インターフェイス	**IEditableCollectionView**, **IPagedCollectionView**

ICollectionView インタフェースを実装して、標準JavaScript配列内の データを公開するクラス。

CollectionView クラスは、次のインタフェースを実装します。

- **ICollectionView** ：現在のレコードの管理、カスタムソート、 フィルタ処理、およびグループ化を提供します。
- **IEditableCollectionView** ：項目を編集、追加、および削除するための メソッドを提供します。
- **IPagedCollectionView** ：ページングを提供します。

CollectionView クラスを使用するには、最初にクラスを宣言し、標準の配列を データソースとして渡します。次に、 **filter**、**sortDescriptions**、**groupDescriptions**、**pageSize** の各プロパティを使用して ビューを構成します。最後に、**items** プロパティを使用してビューにアクセスします。次に例を示します。

```
// 新しいCollectionViewを作成します
var cv = new wijmo.collections.CollectionView(myArray);

// amountで項目を降順にソートします
var sd = new wijmo.collections.SortDescription('amount', false);
cv.sortDescriptions.push(sd);

// amountが100より大きい項目だけを表示します
cv.filter = function(item) { return item.amount > 100 };

// ソートおよびフィルタ処理した結果をコンソールに表示します
for (var i = 0; i < cv.items.length; i++) {
  var item = cv.items[i];
  console.log(i + ': ' + item.name + ' ' + item.amount);
}
```

次の例では、**CollectionView** を使用して、一部の生データをソートする方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/2g70rkct)

コンストラクタ

- constructor

プロパティ

- canAddNew
- canCancelEdit
- canChangePage
- canFilter
- canGroup
- canRemove
- canSort
- currentAddItem
- currentEditItem
- currentItem
- currentPosition
- filter
- getError
- groupDescriptions
- groups

- groups
- isAddingNew
- isEditingItem
- isEmpty
- isPageChanging
- isUpdating
- itemCount
- items
- itemsAdded
- itemsEdited
- itemsRemoved
- newItemCreator
- pageCount
- pageIndex
- pageSize
- sortComparer
- sortConverter
- sortDescriptions
- sortNullsFirst
- sourceCollection
- totalItemCount
- trackChanges
- useStableSort

メソッド

- addNew
- beginUpdate
- cancelEdit
- cancelNew
- clearChanges
- commitEdit
- commitNew
- contains
- deferUpdate
- editItem
- endUpdate
- getAggregate
- implementsInterface
- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage
- onCollectionChanged

- onCollectionChanged
- onCurrentChanged
- onCurrentChanging
- onPageChanged
- onPageChanging
- onSourceCollectionChanged
- onSourceCollectionChanging
- refresh
- remove
- removeAt

イベント

- collectionChanged
- currentChanged
- currentChanging
- pageChanged
- pageChanging
- sourceCollectionChanged
- sourceCollectionChanging

コンストラクタ

constructor

```
constructor(sourceCollection?: any, options?: any): CollectionView
```

CollectionView クラスの新しいインスタンスを初期化します。

パラメーター

- **sourceCollection**: **any** OPTIONAL
  この**CollectionView** のソースとする配列。

- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **CollectionView**

プロパティ

- canAddNew

コレクションに新しい項目を追加できるかどうかを示す値を取得します。

型 **boolean**

- canCancelEdit

適用前の変更を破棄して編集されたオブジェクトの元の値を復元できるかどうかを示す値を取得します。

型 **boolean**

- canChangePage

pageIndex 値を変更できるかどうかを示す値を取得します。

型 **boolean**

- canFilter

このビューが**filter** プロパティによってフィルタリングをサポートしているかどうかを示す値を取得します。

型 **boolean**

- canGroup

このビューが**groupDescriptions** プロパティによってグループ化をサポートしているかどうかを示す値を取得します。

型 **boolean**

- canRemove

コレクションから項目を削除できるかどうかを示す値を取得します。

型 **boolean**

- canSort

このビューが**sortDescriptions** プロパティによってソートをサポートしているかどうかを示す値を取得します。

型 **boolean**

- currentAddItem

現在の追加トランザクションの間に追加される項目を取得します。

型 **any**

- currentEditItem

現在の編集トランザクションの間に編集される項目を取得します。

型 **any**

- currentItem

ビューの現在の項目を取得します。

型 **any**

- currentPosition

ビューの現在の項目の順序位置を取得します。

型 **number**

filter

項目がビューに含める対象として適しているかどうかを判断するために使用されるコールバックを取得または設定します。

このコールバック関数がtrueを返した場合、パラメーターとして渡された項目はビューに含まれます。

メモ: フィルタ関数でスコープ（すなわち、有効な'this'値）が必要な場合は、'this'オブジェクトを指定した'bind'関数を使用してフィルタを設定します。例:

```
collectionView.filter = this._filter.bind(this);
```

型 **IPredicate**

getError

項目の特定のプロパティに検証エラーが含まれているかどうかを判定するコールバックを取得または設定します。

指定すると、コールバックは、検証する項目とプロパティを含む2つの パラメータを受け取り、エラーを説明する文字列を返します （エラーがない場合はnull）。

次に例を示します。

```
var view = new wijmo.collections.CollectionView(data, {
    getError: function (item, property) {
        switch (property) {
            case 'country':
                return countries.indexOf(item.country) < 0
                    ? 'Invalid Country'
                    : null;
            case 'downloads':
            case 'sales':
            case 'expenses':
                return item[property] < 0
                    ? '負にはできません。'
                    : null;
            case 'active':
                return item.active && item.country.match(/US|UK/)
                    ? 'USまたはUKではアクティブな項目が許可されません。'
                    : null;
        }
        return null;
    }
});
```

型 **Function**

groupDescriptions

コレクションの項目をビューでどのようにグループ化するかを記述する **GroupDescription** オブジェクトのコレクションを取得します。

型 **ObservableArray**

groups

最上位レベルのグループを表す **CollectionViewGroup** オブジェクトの配列を取得します。

型 **CollectionViewGroup**[]

isAddingNew

追加トランザクションが進行中であるかどうかを示す値を取得します。

型 **boolean**

- isEditingItem

編集トランザクションが進行中であるかどうかを示す値を取得します。

型 **boolean**

- isEmpty

このビューに項目が1つも含まれていないかどうかを示す値を取得します。

型 **boolean**

- isPageChanging

ページインデックスが変更されているかどうかを示す値を取得します。

型 **boolean**

- isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

型

- itemCount

ページングを適用する前のビューの既知の項目の数を取得します。

型 **number**

- items

ビューの項目を取得します。

型 **any**[]

- itemsAdded

trackChanges が有効化されてから、コレクションに追加されたレコードを含む**ObservableArray** を取得します。

型 **ObservableArray**

- itemsEdited

trackChanges が有効化されてから、コレクションで編集されたレコードを含む**ObservableArray** を取得します。

型 **ObservableArray**

- itemsRemoved

trackChanges が有効化されてから、コレクションから削除されたレコードを含む**ObservableArray** を取得します。

型 **ObservableArray**

- newItemCreator

コレクションの新しい項目を作成する関数を取得または設定します。

作成関数が提供されない場合、**`CollectionView`** は、 適切な型の項目を初期化せずに作成しようとします。

作成関数が提供される場合、その関数は、パラメータを受け取らず、 コレクションに対して適切な型のオブジェクトを初期化して返す 必要があります。

型 **Function**

- pageCount

総ページ数を取得します。

型 **number**

- pageIndex

現在のページの0から始まるインデックスを取得します。

型 **number**

- pageSize

1ページに表示する項目数を取得または設定します。

型 **number**

sortComparer

ソート時に値の比較に使用する関数を取得または設定します。

指定された場合、ソート比較関数は、パラメータとして任意の型の値を 2つ取り、最初の値が2番目の値と比べて小さい、等しい、または大きい のいずれであるかを示す値-1、0、または+1を返します。ソート比較関数がnullを返す場合は、標準の組み込み 比較子が使用されます。

この**sortComparer** プロパティを使用すると、カスタム比較アルゴリズムを 提供でき、単純な文字列比較より、ユーザーが期待する結果によく 一致するソートシーケンスが得られる場合があります。

たとえば、 **Dave Koeleの英数字アルゴリズム**があります。 このアルゴリズムは、文字列を文字列や数値から成る部分に分割した後、 数値部分は値順に、文字列部分はASCII順にソートします。 Daveは、この結果を「自然なソート順」と呼んでいます。

次の例は、**sortComparer** プロパティの一般的な使用方法を示します。

```
// カスタムソート比較子を使用してCollectionViewを作成します
var dataCustomSort = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b)
      ? alphanum(a, b) // 文字列に使用するカスタム比較子
      : null; // 文字列以外にはデフォルトの比較子を使用します
  }
});
```

次の例は、 **Intl.Collator** を使用してソート順を制御する方法を示しています。

```
// Intl.Collatorを使用してソートするCollectionViewを作成します
var collator = window.Intl ? new Intl.Collator() : null;
var dataCollator = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b) && collator
      ? collator.compare(a, b) // 文字列に使用するカスタム比較子
      : null; // 文字列以外にはデフォルトの比較子を使用します
  }
});
```

型 **Function**

sortConverter

ソート時の値の変換に使用される関数を取得または設定します。

この関数は、**SortDescription**、データ項目、および変換する値をパラメーターとして受け取り、変換後の値を返す必要があります。

このプロパティはソートの動作をカスタマイズする手段を提供します。たとえば、**FlexGrid** コントロールはこのプロパティを使用して、マップされた列を生の値ではなく表示値を基準にソートします。

以下のサンプルコードは、国コードの整数を含む'country'プロパティをソートするときに、 対応する国名を使用してソートされるようにします。

```
var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(',');
collectionView.sortConverter = function (sd, item, value) {
  if (sd.property == 'countryMapped') {
    value = countries[value]; // 国IDを国名に変換します。
  }
  return value;
}
```

型 **Function**

sortDescriptions

コレクションの項目をビューでどのようにソートするかを記述する**SortDescription** オブジェクトの配列を取得します。

型 **ObservableArray**

- sortNullsFirst

Gets or sets a value that determines whether null values should appear first or last when the collection is sorted (regardless of sort direction).

This property is false by default, which causes null values to appear last on the sorted collection. This is also the default behavior in Excel.

型 **boolean**

- sourceCollection

基になる（フィルタリングもソートもされていない）コレクションを取得または設定します。

型 **any**

- totalItemCount

ページングを適用する前のビュー内の項目の合計数を取得します。

型 **number**

- trackChanges

コントロールがデータの変更を追跡するかどうかを決定する値を取得または設定します。

trackChanges がtrueに設定されている場合、**CollectionView** は、データの変更を追跡し、 **itemsAdded**、**itemsRemoved**、**itemsEdited** の各コレクションを介して変更を公開します。

変更の追跡は、変更が有効であることをユーザーが確認した後にサーバーを更新する必要がある場合に役立ちます。

変更をコミットまたはキャンセルしたら、**clearChanges** メソッドを使用して、 **itemsAdded**、**itemsRemoved**、**itemsEdited** の各コレクションをクリアします。

CollectionView は、適切な**CollectionView** メソッド（**editItem** /**commitEdit**、 **addNew** /**commitNew**、**remove**）を使用して行われた変更だけを追跡します。 データに直接加えた変更は追跡されません。

型 **boolean**

- useStableSort

安定したソートアルゴリズムを使用するかどうかを取得または設定します。

安定したソートアルゴリズムは、同じキーを持つレコード間の相対的な順序を維持します。 たとえば、"Amount"フィールドを持つオブジェクトのコレクションを考えてみます。 このコレクションを"Amount"でソートする場合、安定したソートでは、Amount値が同じレコード間で元の順序が保たれます。

このプロパティのデフォルトはfalseです。この場合は、高速だが安定ではない JavaScriptの組み込みソートメソッドが**CollectionView** で使用されます。**useStableSort** をtrueに設定すると、ソート時間が30%～50%も長くなります。 コレクションが大きいと、かなりの時間の増加になります。

型 **boolean**

メソッド

addNew

```
addNew(): any
```

新しい項目を作成し、コレクションに追加します。

このメソッドは、パラメータを受け取りません。新しい項目が**commitNew** メソッドでコミットされるか、 **cancelNew** メソッドでキャンセルされるまで、 リフレッシュ操作を保留します。

次のコードは、**addNew** メソッドの典型的な使用方法を示します。

```
// 新しい項目を作成し、それをコレクションに追加します
var newItem = view.addNew();

// 新しい項目を初期化します
newItem.id = getFreshId();
newItem.name = '新しい顧客';

// ビューをリフレッシュできるように新しい項目をコミットします
view.commitNew();
```

新しい項目を**sourceCollection** にプッシュしてから、**refresh** メソッドを 呼び出すことで、新しい項目を追加することもできます。**addNew** では、 ユーザーが追加操作をキャンセルできるため、ユーザー対話式のシナリオ （データグリッドに新しい項目の追加するなど）で特に便利です。また、追加操作がコミットされるまで、新しい項目がソートされたり、 フィルタ処理でビューから除外されないようにします。

戻り値 **any**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで更新を中断します。

戻り値 **void**

cancelEdit

```
cancelEdit(): void
```

現在の編集トランザクションを終了し、可能であれば項目を元の値に戻します。

戻り値 **void**

cancelNew

```
cancelNew(): void
```

現在の追加トランザクションを終了し、追加前の新しい項目を破棄します。

戻り値 **void**

clearChanges

```
clearChanges(): void
```

itemsAdded、**itemsRemoved**、**itemsEdited** の各コレクションの全項目をクリアすることによってすべての変更をクリアします。

このメソッドは、変更をサーバーに確定した後またはデータをサーバーから更新した後に呼び出します。

戻り値 **void**

commitEdit

```
commitEdit(): void
```

現在の編集トランザクションを終了し、適用前の変更を保存します。

戻り値 **void**

commitNew

```
commitNew(): void
```

現在の追加トランザクションを終了し、追加前の新しい項目を保存します。

戻り値 **void**

contains

```
contains(item: any): boolean
```

指定した項目がこのビューに属するかどうかを示す値を返します。

パラメーター

- **item**: **any**
  調べる項目。

戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

戻り値 **void**

editItem

```
editItem(item: any): void
```

指定した項目の編集トランザクションを開始します。

パラメーター

- **item**: **any**
  編集する項目。

戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された更新を再開します。

戻り値 **void**

getAggregate

```
getAggregate(aggType: Aggregate, binding: string, currentPage?: boolean): void
```

このコレクション内の項目の集計値を計算します。

パラメーター

- **aggType**: **Aggregate**
  計算する集計のタイプ。
- **binding**: **string**
  集計するプロパティ。
- **currentPage**: **boolean** OPTIONAL
  現在のページの項目だけを含めるかどうか。

戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

戻り値 **boolean**

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

指定した項目をビューの現在の項目に設定します。

パラメーター

- **item**: **any**
  現在の項目として設定する項目。

戻り値 **boolean**

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

ビューの最初の項目を現在の項目として設定します。

戻り値 **boolean**

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

ビューの最後の項目を現在の項目として設定します。

戻り値 **boolean**

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

ビューの現在の項目の後の項目を現在の項目として設定します。

戻り値 **boolean**

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

ビューの指定したインデックスにある項目を現在の項目として設定します。

パラメーター

- **index**: **number**
  現在の項目として設定する項目のインデックス。

戻り値 **boolean**

moveCurrentToPrevious

```
moveCurrentToPrevious(): boolean
```

ビューの現在の項目の前の項目を現在の項目として設定します。

戻り値 **boolean**

moveToFirstPage

```
moveToFirstPage(): boolean
```

最初のページを現在のページとして設定します。

戻り値 **boolean**

moveToLastPage

```
moveToLastPage(): boolean
```

最後のページを現在のページとして設定します。

戻り値 **boolean**

moveToNextPage

```
moveToNextPage(): boolean
```

現在のページの後のページに移動します。

戻り値 **boolean**

moveToPage

```
moveToPage(index: number): boolean
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先ページのインデックス。

戻り値 **boolean**

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

現在のページの前のページに移動します。

戻り値 **boolean**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

collectionChanged イベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  変更の記述が含まれます。

戻り値 **void**

onCurrentChanged

```
onCurrentChanged(e?: EventArgs): void
```

currentChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onCurrentChanging

```
onCurrentChanging(e: CancelEventArgs): boolean
```

currentChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs**。

戻り値 **boolean**

onPageChanged

```
onPageChanged(e?: EventArgs): void
```

pageChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onPageChanging

```
onPageChanging(e: PageChangingEventArgs): boolean
```

pageChanging イベントを発生させます。

パラメーター

- **e**: **PageChangingEventArgs**
  イベントデータを含む**PageChangingEventArgs**。

戻り値 **boolean**

onSourceCollectionChanged

```
onSourceCollectionChanged(e?: EventArgs): void
```

sourceCollectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onSourceCollectionChanging

```
onSourceCollectionChanging(e: CancelEventArgs): boolean
```

sourceCollectionChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs**。

戻り値 **boolean**

refresh

```
refresh(): void
```

現在のソート、フィルタ、およびグループパラメーターを使用してビューを再作成します。

戻り値 **void**

remove

```
remove(item: any): void
```

指定した項目をコレクションから削除します。

パラメーター

- **item**: **any**
  コレクションから削除する項目。

戻り値 **void**

removeAt

```
removeAt(index: number): void
```

指定したインデックスにある項目をコレクションから削除します。

パラメーター

- **index**: **number**
  コレクションから削除する項目のインデックス。 このインデックスは、ソースコレクションに対してではなくビューに対する相対インデックスです。

戻り値 **void**

イベント

collectionChanged

コレクションが変更されたときに発生します。

引数 **NotifyCollectionChangedEventArgs**

currentChanged

現在の項目が変更された後に発生します。

引数 **EventArgs**

currentChanging

現在の項目が変更される前に発生します。

引数 **CancelEventArgs**

pageChanged

ページインデックスが変更された後に発生します。

引数 **EventArgs**

pageChanging

ページインデックスが変更される前に発生します。

引数 **PageChangingEventArgs**

sourceCollectionChanged

sourceCollection プロパティの値が変化した後に発生します。

引数 **EventArgs**

sourceCollectionChanging

sourceCollection プロパティの値が変化する前に発生します。

引数	CancelEventArgs

CollectionViewGroup クラス

ファイル wijmo.js
モジュール **wijmo.collections**

CollectionView オブジェクトの**groupDescriptions** プロパティに基づいて作成されたグループを表します。

コンストラクタ

- constructor

メソッド

- getAggregate

コンストラクタ

constructor

```
constructor(groupDescription: GroupDescription, name: string, level: number, isBottomLevel: boolean): CollectionViewGroup
```

CollectionViewGroup クラスの新しいインスタンスを初期化します。

パラメーター

- **groupDescription**: **GroupDescription**
  新しいグループを所有する **GroupDescription**。
- **name**: **string**
  新しいグループの名前。
- **level**: **number**
  新しいグループのレベル。
- **isBottomLevel**: **boolean**
  このグループがサブグループを持つかどうか。

戻り値 **CollectionViewGroup**

メソッド

- getAggregate

```
getAggregate(aggType: Aggregate, binding: string, view?: ICollectionView): void
```

このグループの項目の集計値を計算します。

パラメーター

- **aggType**: **Aggregate**
  計算する集計のタイプ。
- **binding**: **string**
  集計するプロパティ。
- **view**: **ICollectionView** OPTIONAL
  このグループを所有するCollectionView。

戻り値 **void**

GroupDescription クラス

ファイル wijmo.js
モジュール **wijmo.collections**
派生クラス **PropertyGroupDescription**

グループ化条件を定義する型の基本クラスを表します。

この目的に一般に使用される具象クラスは**PropertyGroupDescription** です。

メソッド

- groupNameFromItem
- namesMatch

メソッド

groupNameFromItem

```
groupNameFromItem(item: any, level: number): any
```

指定した項目のグループ名を返します。

パラメーター

- **item**: **any**
  グループ名を取得する項目。
- **level**: **number**
  0から始まるグループレベルインデックス。

戻り値 **any**

namesMatch

```
namesMatch(groupName: any, itemName: any): boolean
```

グループ名と項目名が一致する（これは項目がグループに属していることを意味します）かどうかを示す値を返します。

パラメーター

- **groupName**: **any**
  グループの名前。
- **itemName**: **any**
  項目の名前。

戻り値 **boolean**

NotifyCollectionChangedEventArgs クラス

ファイル wijmo.js
モジュール **wijmo.collections**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

collectionChanged イベントのデータを提供します。

コンストラクタ

- constructor

プロパティ

- action
- empty
- index
- item
- reset

コンストラクタ

constructor

```
constructor(action?: NotifyCollectionChangedAction, item?, index?: number): NotifyCollectionChangedEventArgs
```

NotifyCollectionChangedEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **action**: **NotifyCollectionChangedAction** OPTIONAL
  イベントを発生させたアクションのタイプ。
- **item**: OPTIONAL
  追加または変更された項目。
- **index**: **number** OPTIONAL
  項目のインデックス。

戻り値 **NotifyCollectionChangedEventArgs**

プロパティ

- action

イベントを発生させたアクションを取得します。

型 **NotifyCollectionChangedAction**

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

- index

変更が発生したインデックスを取得します。

型 **number**

- item

追加、削除、または変更された項目を取得します。

型 **any**

- STATIC reset

リセット通知を提供します。

型 **NotifyCollectionChangedEventArgs**

ObservableArray クラス

ファイル	`wijmo.js`
モジュール	**wijmo.collections**
基本クラス	**ArrayBase**
派生クラス	**RowColCollection**, **AxisCollection**, **PlotAreaCollection**, **SheetCollection**, **PivotFieldCollection**
インターフェイス	**INotifyCollectionChanged**
表示	☐継承されたメンバー ☐イベント発生元

変更に関する通知を送信する配列。

このクラスは、push、pop、splice、insert、removeのいずれかのメソッドによって変更が加えられたときに**collectionChanged** イベントを発生させます。

警告: 配列メンバに値を直接割り当てることによって配列を変更したとき、または配列の長さを変更したときには、**collectionChanged** イベントは発生しません。

コンストラクタ

- constructor

プロパティ

- isUpdating

メソッド

- beginUpdate
- clear
- deferUpdate
- endUpdate
- implementsInterface
- indexOf
- insert
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

イベント

- collectionChanged

コンストラクタ

constructor

```
constructor(data?: any[]): ObservableArray
```

ObservableArray クラスの新しいインスタンスを初期化します。

パラメーター

- **data**: **any**[] OPTIONAL
  ObservableArray に格納する項目を含む配列。

戻り値 **ObservableArray**

プロパティ

- isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

型

メソッド

- beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

戻り値 **void**

- clear

```
clear(): void
```

配列からすべての項目を削除します。

戻り値 **void**

- deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

戻り値 **boolean**

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

配列内で項目を検索します。

パラメーター

- **searchElement**: **any**
  配列内で検索する要素。

- **fromIndex**: **number** OPTIONAL
  検索を開始するインデックス。

戻り値 **number**

insert

```
insert(index: number, item: any): void
```

配列内の指定した位置に項目を挿入します。

パラメーター

- **index**: **number**
  項目を追加する位置。

- **item**: **any**
  配列に追加する項目。

戻り値 **void**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

collectionChanged イベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  変更の記述が含まれます。

戻り値 **void**

push

```
push(...item: any[]): number
```

配列の末尾に1つ以上の項目を追加します。

パラメーター

- **...item**: **any[]**
  配列に追加する1つ以上の項目。

戻り値 **number**

remove

```
remove(item: any): boolean
```

配列から項目を削除します。

パラメーター

- **item**: **any**
  削除する項目。

戻り値 **boolean**

removeAt

```
removeAt(index: number): void
```

配列内の指定した位置にある項目を削除します。

パラメーター

- **index**: **number**
  削除する項目の位置。

戻り値 **void**

setAt

```
setAt(index: number, item: any): void
```

配列内の指定した位置にある項目を設定します。

パラメーター

- **index**: **number**
  項目を設定する位置。

- **item**: **any**
  配列に設定する項目。

戻り値 **void**

slice

```
slice(begin?: number, end?: number): any[]
```

配列の一部のシャローコピーを作成します。

パラメーター

- **begin**: **number** OPTIONAL
  コピーを開始する位置。

- **end**: **number** OPTIONAL
  コピーを終了する位置。

戻り値 **any[]**

sort

```
sort(compareFn?: Function): this
```

配列の要素を適切な位置にソートします。

パラメーター

- **compareFn**: **Function** OPTIONAL
  ソート順序を定義する関数。 この関数は2つの引数を受け取り、-1（最初の引数の方が2番目の引数より小さい場合）、+1（大きい場合）、0（等しい場合）のいずれかの値を返す必要があります。 これを省略した場合は、各要素の文字列変換に従って辞書順にソートされます。

戻り値 **this**

splice

```
splice(index: number, count: number, item?: any): any[]
```

配列からの項目の削除と配列への項目の追加の一方または両方を行います。

パラメーター

- **index**: **number**
  項目を追加または削除する位置。
- **count**: **number**
  配列から削除する項目の数。
- **item**: **any** OPTIONAL
  配列に追加する項目。

戻り値 **any[]**

イベント

collectionChanged

コレクションが変更されたときに発生します。

引数 **NotifyCollectionChangedEventArgs**

PageChangingEventArgs クラス

ファイル wijmo.js
モジュール **wijmo.collections**
基本クラス **CancelEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

pageChanging イベントのデータを提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- empty
- newPageIndex

コンストラクタ

constructor

```
constructor(newIndex: number): PageChangingEventArgs
```

PageChangingEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **newIndex**: **number**
  現在のページとして設定されるページのインデックス。

戻り値 **PageChangingEventArgs**

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

- newPageIndex

現在のページとして設定されるページのインデックスを取得します。

型 **number**

PropertyGroupDescription クラス

ファイル wijmo.js
モジュール **wijmo.collections**
基本クラス **GroupDescription**
表示 ☐継承されたメンバー ☐イベント発生元

プロパティ名を基準として項目のグループ化を記述します。

たとえば、以下のコードは**CollectionView** の項目を項目の'country'プロパティの値に従ってグループ化します。

```
var cv = new wijmo.collections.CollectionView(items);
var gd = new wijmo.collections.PropertyGroupDescription('country');
cv.groupDescriptions.push(gd);
```

また、グループ名を生成するコールバック関数を指定することもできます。 たとえば、以下のコードは **CollectionView** の項目を項目の'country'プロパティの値の最初の文字に従ってグループ化します。

```
var cv = new wijmo.collections.CollectionView(items);
var gd = new wijmo.collections.PropertyGroupDescription('country',
  function(item, propName) {
    return item[propName][0]; // 国の最初の文字を返します。
});
cv.groupDescriptions.push(gd);
```

コンストラクタ

- constructor

プロパティ

- propertyName

メソッド

- groupNameFromItem
- namesMatch

コンストラクタ

constructor

```
constructor(property: string, converter?: Function): PropertyGroupDescription
```

PropertyGroupDescription クラスの新しいインスタンスを初期化します。

パラメーター

- **property**: **string**
  項目が属するグループを指定するプロパティの名前。

- **converter**: **Function** OPTIONAL
  項目とプロパティ名を受け取ってグループ名を返すコールバック関数。 これを指定しない場合、グループ名は項目のプロパティ値になります。

戻り値 **PropertyGroupDescription**

プロパティ

propertyName

項目が属するグループを決定するのに使用されるプロパティの名前を取得します。

型 **string**

メソッド

groupNameFromItem

```
groupNameFromItem(item: any, level: number): any
```

指定した項目のグループ名を返します。

パラメーター

- **item**: **any**
  グループ名を取得する項目。
- **level**: **number**
  0から始まるグループレベルインデックス。

戻り値 **any**

namesMatch

```
namesMatch(groupName: any, itemName: any): boolean
```

グループ名と項目名が一致する（これは項目がグループに属していることを意味します）かどうかを示す値を返します。

パラメーター

- **groupName**: **any**
  グループの名前。
- **itemName**: **any**
  項目の名前。

戻り値 **boolean**

SortDescription クラス

ファイル wijmo.js
モジュール **wijmo.collections**

ソート基準を記述します。

コンストラクタ

- constructor

プロパティ

- ascending
- property

コンストラクタ

constructor

```
constructor(property: string, ascending: boolean): SortDescription
```

SortDescription クラスの新しいインスタンスを初期化します。

パラメーター

- **property**: **string**
  ソートの基準となるプロパティの名前。
- **ascending**: **boolean**
  昇順でソートするかどうか。

戻り値 **SortDescription**

プロパティ

- ascending

値を昇順にソートするかどうかを示す値を取得します。

型 **boolean**

property

ソートに使用されるプロパティの名前を取得します。

型 **string**

ICollectionView インターフェイス

ファイル `wijmo.js`
モジュール **wijmo.collections**
インターフェイス **INotifyCollectionChanged**

コレクションで現在レコード管理、カスタムのソート、フィルタリング、およびグループ化の機能を実現します。

これはMicrosoftのXAMLプラットフォームで使用される**ICollectionView**インタフェースのJavaScript版です。データをUI要素にバインドするための、MVVMを実装しやすい強力で一貫性のある手段を提供します。

Wijmoには、**ICollectionView** を実装するクラスがいくつか用意されています。最もよく使用するものは **CollectionView** で、これは通常のJavaScript配列に基づいて機能します。

プロパティ

- canFilter
- canGroup
- canSort
- currentChanged
- currentChanging
- currentItem
- currentPosition
- filter
- groupDescriptions
- groups
- isEmpty
- items
- sortDescriptions
- sourceCollection

メソッド

- beginUpdate
- contains
- deferUpdate
- endUpdate
- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- refresh

プロパティ

- canFilter

このビューが**`filter`** プロパティによってフィルタリングをサポートしているかどうかを示す値を取得します。

型 **boolean**

- canGroup

このビューが**groupDescriptions** プロパティによってグループ化をサポートしているかどうかを示す値を取得します。

型 **boolean**

- canSort

このビューが**sortDescriptions** プロパティによってソートをサポートしているかどうかを示す値を取得します。

型 **boolean**

- currentChanged

現在の項目が変更された後に発生します。

型 **Event**

- currentChanging

現在の項目が変更される前に発生します。

型 **Event**

- currentItem

ビューの現在の項目を取得します。

型 **any**

- currentPosition

ビューの現在の項目の順序位置を取得します。

型 **number**

- filter

項目がビューに含める対象として適しているかどうかを判断するために使用されるコールバックを取得または設定します。

メモ: フィルタ関数でスコープ（すなわち、有効な'this'値）が必要な場合は、'this'オブジェクトを指定した'bind'関数を使用してフィルタを設定します。例:

```
collectionView.filter = this._filter.bind(this);
```

型 **IPredicate**

- groupDescriptions

コレクションの項目をビューでどのようにグループ化するかを記述する **GroupDescription** オブジェクトのコレクションを取得します。

型 **ObservableArray**

- groups

最上位レベルのグループを取得します。

型 **any[]**

- isEmpty

このビューに項目が1つも含まれていないかどうかを示す値を取得します。

型 **boolean**

- items

フィルタリング、ソート、グループ化が適用されたビューの項目を取得します。

型 **any[]**

- sortDescriptions

コレクションの項目をビューでどのようにソートするかを記述する**SortDescription** オブジェクトのコレクションを取得します。

型 **ObservableArray**

- sourceCollection

このビューの作成元のコレクションオブジェクトを取得または設定します。

型 **any**

メソッド

- beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで更新を中断します。

戻り値 **void**

- contains

```
contains(item: any): boolean
```

指定した項目がこのビューに属するかどうかを示す値を返します。

パラメーター

- **item**: **any**
  コレクション内で検索する項目。

戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコレクションは更新されません。 このメソッドは、関数が例外をスローした場合でもendUpdateが確実に呼び出されるようにします。

パラメーター

- **fn**: **Function**
  beginUpdate/endUpdateブロック内で実行する関数。

戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された更新を再開します。

戻り値 **void**

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

指定した項目をビューの現在の項目に設定します。

パラメーター

- **item**: **any**
  currentItem として設定する項目。

戻り値 **boolean**

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

ビューの最初の項目を現在の項目として設定します。

戻り値 **boolean**

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

ビューの最後の項目を現在の項目として設定します。

戻り値 **boolean**

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

ビューの現在の項目の後の項目を現在の項目として設定します。

戻り値 **boolean**

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

ビューの指定したインデックスにある項目を現在の項目として設定します。

パラメーター

- **index**: **number**
  `currentItem` として設定する項目のインデックス。

戻り値 **boolean**

moveCurrentToPrevious

```
moveCurrentToPrevious(): void
```

ビューの現在の項目の前の項目を現在の項目として設定します。

戻り値 **void**

refresh

```
refresh(): void
```

現在のソート、フィルタ、およびグループパラメーターを使用してビューを再作成します。

戻り値 **void**

IComparer インターフェイス

ファイル wijmo.js
モジュール **wijmo.collections**

2つのオブジェクトを比較するメソッドを表します。

IEditableCollectionView インターフェイス

ファイル wijmo.js
モジュール **wijmo.collections**
インターフェイス **ICollectionView**

ICollectionView を拡張して編集機能を提供するメソッドとプロパティを定義します。

プロパティ

- canAddNew
- canCancelEdit
- canRemove
- currentAddItem
- currentEditItem
- isAddingNew
- isEditingItem

メソッド

- addNew
- cancelEdit
- cancelNew
- commitEdit
- commitNew
- editItem
- remove
- removeAt

プロパティ

canAddNew

コレクションに新しい項目を追加できるかどうかを示す値を取得します。

型 **boolean**

canCancelEdit

適用前の変更を破棄して編集されたオブジェクトの元の値を復元できるかどうかを示す値を取得します。

型 **boolean**

canRemove

コレクションから項目を削除できるかどうかを示す値を取得します。

型 **boolean**

currentAddItem

現在の追加トランザクションの間に追加される項目を取得します。

型 **any**

- currentEditItem

現在の編集トランザクションの間に編集される項目を取得します。

型 **any**

- isAddingNew

追加トランザクションが進行中であるかどうかを示す値を取得します。

型 **boolean**

- isEditingItem

編集トランザクションが進行中であるかどうかを示す値を取得します。

型 **boolean**

メソッド

- addNew

```
addNew(): any
```

コレクションに新しい項目を追加します。

戻り値 **any**

- cancelEdit

```
cancelEdit(): void
```

現在の編集トランザクションを終了し、可能であれば項目を元の値に戻します。

戻り値 **void**

- cancelNew

```
cancelNew(): void
```

現在の追加トランザクションを終了し、追加前の新しい項目を破棄します。

戻り値 **void**

- commitEdit

```
commitEdit(): void
```

現在の編集トランザクションを終了し、適用前の変更を保存します。

戻り値 **void**

commitNew

```
commitNew(): void
```

現在の追加トランザクションを終了し、追加前の新しい項目を保存します。

戻り値 **void**

editItem

```
editItem(item: any): void
```

指定した項目の編集トランザクションを開始します。

パラメーター

- **item**: **any**
  編集する項目。

戻り値 **void**

remove

```
remove(item: any): void
```

指定した項目をコレクションから削除します。

パラメーター

- **item**: **any**
  コレクションから削除する項目。

戻り値 **void**

removeAt

```
removeAt(index: number): void
```

指定したインデックスにある項目をコレクションから削除します。

パラメーター

- **index**: **number**
  コレクションから削除する項目のインデックス。

戻り値 **void**

INotifyCollectionChanged インターフェイス

ファイル wijmo.js
モジュール **wijmo.collections**

項目が追加または削除されたときや、コレクションがソート、フィルタリング、グループ化されたときなどに、動的な変更をリスナーに通知します。

プロパティ

- collectionChanged

プロパティ

- collectionChanged

コレクションが変更されたときに発生します。

型 **Event**

IPagedCollectionView インターフェイス

ファイル `wijmo.js`
モジュール **wijmo.collections**
インターフェイス **ICollectionView**

ICollectionView を拡張してページング機能を提供するメソッドとプロパティを定義します。

プロパティ

- canChangePage
- isPageChanging
- itemCount
- pageChanged
- pageChanging
- pageIndex
- pageSize
- totalItemCount

メソッド

- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage

プロパティ

canChangePage

`pageIndex` 値を変更できるかどうかを示す値を取得します。

型 **boolean**

isPageChanging

インデックスが変更されているかどうかを示す値を取得します。

型 **boolean**

itemCount

ページングを考慮してビューの項目数を取得します。

項目の総数を取得するには、**`totalItemCount`** プロパティを使用します。

これは、.NET **IPagedCollectionView**とは異なります。 .NETの**itemCount**と**totalItemCount**はどちらも、 ページングを適用する前の数を返します。

型 **number**

- pageChanged

ページインデックスが変更された後に発生します。

型 **Event**

- pageChanging

ページインデックスが変更される前に発生します。

型 **Event**

- pageIndex

現在のページの0から始まるインデックスを取得します。

型 **number**

- pageSize

1ページに表示する項目数を取得または設定します。

型 **number**

- totalItemCount

ページングを適用する前のビュー内の項目の合計数を取得します。

ページングを考慮せずに現在のビュー内の項目の数を取得するには、 **itemCount** プロパティを使用します。

これは、.NET **IPagedCollectionView**とは異なります。 .NETの**itemCount**と**totalItemCount**はどちらも、 ページングを適用する前の数を返します。

型 **number**

メソッド

- moveToFirstPage

```
moveToFirstPage(): boolean
```

最初のページを現在のページとして設定します。

戻り値 **boolean**

- moveToLastPage

```
moveToLastPage(): boolean
```

最後のページを現在のページとして設定します。

戻り値 **boolean**

moveToNextPage

```
moveToNextPage(): boolean
```

現在のページの後のページに移動します。

戻り値 **boolean**

moveToPage

```
moveToPage(index: number): boolean
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先ページのインデックス。

戻り値 **boolean**

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

現在のページの前のページに移動します。

戻り値 **boolean**

IPredicate インターフェイス

ファイル wijmo.js
モジュール **wijmo.collections**

任意の型の項目を受け取り、オブジェクトが基準セットを満たしているかどうかを示すブール値を返すメソッドを表します。

NotifyCollectionChangedAction 列挙体

ファイル wijmo.js
モジュール **wijmo.collections**

collectionChanged イベントを発生させたアクションを記述します。

メンバー

名前	値	説明
Add	0	項目がコレクションに追加されました。
Remove	1	項目がコレクションから削除されました。
Change	2	項目が変更または置換されました。
Reset	3	複数の項目が同時に変更されました（たとえば、コレクションのソート、フィルタリング、またはグループ化が実行されたときなど）。

wijmo.grid モジュール

ファイル `wijmo.grid.js`
モジュール **wijmo.grid**

FlexGrid コントロールおよび関連クラスを定義します。

この例では、**FlexGrid** コントロールを作成し、それを'data'配列に連結しています。グリッドには4つの列があり、これらは、グリッドの**columns** 配列にデータを明示的に挿入することで指定されます。

サンプル

Show me (https://jsfiddle.net/Wijmo5/6GB66)

クラス

- CellEditEndingEventArgs
- CellFactory
- CellRange
- CellRangeEventArgs
- Column
- ColumnCollection
- DataMap
- FlexGrid
- FormatItemEventArgs
- GridPanel
- GroupRow
- HitTestInfo
- MergeManager
- Row
- RowCol
- RowColCollection
- RowCollection

列挙体

- AllowDragging
- AllowMerging
- AllowResizing
- AutoSizeMode
- CellType
- HeadersVisibility
- KeyAction
- RowColFlags
- SelectedState
- SelectionMode
- SelMove

CellEditEndingEventArgs クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
基本クラス **CellRangeEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

cellEditEnding イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- col
- data
- empty
- panel
- range
- refresh
- row
- stayInEditMode

コンストラクタ

constructor

```
constructor(p: GridPanel, rng: CellRange, data?: any): CellRangeEventArgs
```

CellRangeEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。
- **rng**: **CellRange**
  イベントの影響を受けるセルの範囲。
- **data**: **any** OPTIONAL
  このイベントに関連するデータ。

継承元 **CellRangeEventArgs**
戻り値 **CellRangeEventArgs**

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

col

このイベントの影響を受けた列を取得します。

継承元 **CellRangeEventArgs**
型 **number**

data

このイベントに関連するデータを取得または設定します。

継承元 **CellRangeEventArgs**
型 **any**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

panel

このイベントの影響を受ける **GridPanel** を取得します。

継承元 **CellRangeEventArgs**
型 **GridPanel**

range

このイベントの影響を受ける **CellRange** を取得します。

継承元 **CellRangeEventArgs**
型 **CellRange**

refresh

編集が完了した後にグリッドのすべての内容を更新するかどうかを決定する値を取得または設定します。

型 **boolean**

row

このイベントの影響を受けた行を取得します。

継承元 **CellRangeEventArgs**
型 **number**

stayInEditMode

編集を終了するのではなく、セルを編集モードのままにするかどうかを取得または設定します。

型 **boolean**

CellFactory クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**

FlexGrid コントロール内のセルを表すHTML要素を作成します。

メソッド

- disposeCell
- getEditorValue
- updateCell

メソッド

- disposeCell

```
disposeCell(cell: HTMLElement): void
```

セル要素を破棄し、その要素に関連するすべてのリソースを解放します。

パラメーター
- **cell**: **HTMLElement**
  セルを表す要素。

戻り値 **void**

- getEditorValue

```
getEditorValue(g: FlexGrid): any
```

現在使用中のエディタの値を取得します。

パラメーター
- **g**: **FlexGrid**
  エディタを所有する**FlexGrid** 。

戻り値 **any**

updateCell

```
updateCell(p: GridPanel, r: number, c: number, cell: HTMLElement, rng?: CellRange, updateContent?: boolean): void
```

グリッドのセルを作成または更新します。

パラメーター

- **p**: **GridPanel**
  セルを含む**GridPanel** 。
- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **cell**: **HTMLElement**
  セルを表す要素。
- **rng**: **CellRange** OPTIONAL
  セルの結合範囲を含む**CellRange** オブジェクト。セルが結合されていない場合はnull。
- **updateContent**: **boolean** OPTIONAL
  セルのコンテンツと共に 位置とスタイルを更新するかどうか。

戻り値 **void**

CellRange クラス

ファイル `wijmo.grid.js`
モジュール **wijmo.grid**

2つの行インデックスと2つの列インデックスによって定義されたセルの矩形グループを表します。

コンストラクタ

- constructor

プロパティ

- bottomRow
- col
- col2
- columnSpan
- isSingleCell
- isValid
- leftCol
- rightCol
- row
- row2
- rowSpan
- topRow

メソッド

- clone
- contains
- containsColumn
- containsRow
- equals
- getRenderSize
- intersects
- intersectsColumn
- intersectsRow
- setRange

コンストラクタ

constructor

```
constructor(r?: number, c?: number, r2?: number, c2?: number): CellRange
```

CellRange クラスの新しいインスタンスを初期化します。

パラメーター

- **r**: **number** OPTIONAL
  範囲の最初の行のインデックス（デフォルトは-1）。
- **c**: **number** OPTIONAL
  範囲の最初の列のインデックス（デフォルトは-1）。
- **r2**: **number** OPTIONAL
  範囲内の最後の行のインデックス（デフォルトは**r**）。
- **c2**: **number** OPTIONAL
  範囲内の最後の列のインデックス（デフォルトは**r**）。

戻り値 **CellRange**

プロパティ

bottomRow

範囲の一番下の行のインデックスを取得します。

型 **number**

col

範囲内の最初の列のインデックスを取得または設定します。

型 **number**

col2

範囲の2番目の列のインデックスを取得または設定します。

型 **number**

columnSpan

範囲の列数を取得します。

型 **number**

isSingleCell

この範囲が単一のセルに対応するかどうか（最初の行と最後の行のインデックスが同じであり、なおかつ最初の列と最後の列のインデックスが同じであるかどうか）をチェックします。

型 **boolean**

- isValid

範囲に含まれている行インデックスと列インデックスが有効かどうか（行および列の値がゼロ以上かどうか）をチェックします。

型 **boolean**

- leftCol

範囲の一番左の列のインデックスを取得します。

型 **number**

- rightCol

範囲の一番右の列のインデックスを取得します。

型 **number**

- row

範囲内の最初の行のインデックスを取得または設定します。

型 **number**

- row2

範囲の2番目の行のインデックスを取得または設定します。

型 **number**

- rowSpan

範囲の行数を取得します。

型 **number**

- topRow

範囲の一番上の行のインデックスを取得します。

型 **number**

メソッド

- clone

```
clone(): CellRange
```

範囲のコピーを作成します。

戻り値 **CellRange**

contains

```
contains(r: any, c?: number): boolean
```

この範囲が別の範囲または特定のセルを含むかどうかをチェックします。

パラメーター

- **r**: **any**
  調べるCellRangeまたは行インデックス。

- **c**: **number** OPTIONAL
  列インデックス（rパラメーターがCellRangeオブジェクトでない場合に必要）。

戻り値 **boolean**

containsColumn

```
containsColumn(c: number): boolean
```

この範囲が指定した列を含むかどうかをチェックします。

パラメーター

- **c**: **number**
  調べる列のインデックス。

戻り値 **boolean**

containsRow

```
containsRow(r: number): boolean
```

この範囲が指定した行を含むかどうかをチェックします。

パラメーター

- **r**: **number**
  調べる行のインデックス。

戻り値 **boolean**

equals

```
equals(rng: CellRange): boolean
```

この範囲が別の範囲と等しいかどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  この範囲と比較するCellRangeオブジェクト。

戻り値 **boolean**

getRenderSize

```
getRenderSize(p: GridPanel): Size
```

この範囲のレンダリングサイズを取得します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** オブジェクト。

戻り値 **Size**

intersects

```
intersects(rng: CellRange): boolean
```

この範囲が別の範囲と交差するかどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックするCellRangeオブジェクト。

戻り値 **boolean**

intersectsColumn

```
intersectsColumn(rng: CellRange): boolean
```

範囲が別の範囲内の列と交差するかどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックするCellRangeオブジェクト。

戻り値 **boolean**

intersectsRow

```
intersectsRow(rng: CellRange): boolean
```

範囲が別の範囲内の行と交差するかどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックするCellRangeオブジェクト。

戻り値 **boolean**

setRange

```
setRange(r?: number, c?: number, r2?: number, c2?: number): void
```

既存の**CellRange** を初期化します。

パラメーター

- **r**: **number** OPTIONAL
  範囲の最初の行のインデックス（デフォルトは-1）。
- **c**: **number** OPTIONAL
  範囲の最初の列のインデックス（デフォルトは-1）。
- **r2**: **number** OPTIONAL
  範囲内の最後の行のインデックス（デフォルトは**r**）。
- **c2**: **number** OPTIONAL
  範囲内の最後の列のインデックス（デフォルトは**r**）。

戻り値 **void**

CellRangeEventArgs クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
基本クラス **CancelEventArgs**
派生クラス **CellEditEndingEventArgs**, **FormatItemEventArgs**, **XlsxFormatItemEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

CellRange イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- col
- data
- empty
- panel
- range
- row

コンストラクタ

constructor

```
constructor(p: GridPanel, rng: CellRange, data?: any): CellRangeEventArgs
```

CellRangeEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。
- **rng**: **CellRange**
  イベントの影響を受けるセルの範囲。
- **data**: **any** OPTIONAL
  このイベントに関連するデータ。

戻り値 **CellRangeEventArgs**

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

col

このイベントの影響を受けた列を取得します。

型 **number**

data

このイベントに関連するデータを取得または設定します。

型 **any**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

panel

このイベントの影響を受ける **GridPanel** を取得します。

型 **GridPanel**

range

このイベントの影響を受ける **CellRange** を取得します。

型 **CellRange**

row

このイベントの影響を受けた行を取得します。

型 **number**

Column クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
基本クラス **RowCol**
派生クラス **WjFlexGridColumn**
表示 ☐継承されたメンバー ☐イベント発生元

グリッドの列を表します。

コンストラクタ

- constructor

プロパティ

- aggregate
- align
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- binding
- collectionView
- cssClass
- currentSort
- dataMap
- dataType
- describedById
- dropDownCssClass
- format
- grid
- header
- index
- inputType
- isContentHtml
- isReadOnly
- isRequired
- isSelected
- isVisible
- mask
- maxLength
- maxWidth
- minWidth
- multiLine
- name
- pos
- quickAutoSize
- renderSize
- renderWidth
- showDropDown
- size
- sortMemberPath

- sortMemberPath
- visible
- visibleIndex
- width
- wordWrap

メソッド

- getAlignment
- getIsRequired
- onPropertyChanged

コンストラクタ

constructor

```
constructor(options?: any): Column
```

Column クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  列の初期化オプション。

戻り値 **Column**

プロパティ

- aggregate

列のグループヘッダ行に表示する **Aggregate** を取得または設定します。

型 **Aggregate**

- align

列の項目の水平方向の配置を取得または設定します。

このプロパティのデフォルト値はnullで、列の**dataType** に基づいて配置が自動的に選択されます（数値の場合は右揃え、ブール値の場合は中央揃え、その他の型の場合は左揃え）。

デフォルトの配置をオーバーライドする場合は、 このプロパティを'left'、'right'、'center'のいずれかに設定します。

型 **string**

- allowDragging

ユーザーがマウスで行または列を新しい位置に移動できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- allowMerging

行または列にあるセルを結合できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- allowResizing

ユーザーがマウスで行または列をサイズ変更できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- allowSorting

ユーザーがヘッダーをクリックして列をソートできるかどうかを示す値を取得または設定します。

型 **boolean**

- binding

列がバインドされているプロパティの名前を取得または設定します。

型 **string**

- collectionView

この行または列にバインドされた**ICollectionView** を取得します。

継承元 **RowCol**
型 **ICollectionView**

- cssClass

行または列のヘッダ以外のセルをレンダリングするときに使用するCSSクラス名を取得または設定します。

継承元 **RowCol**
型 **string**

- currentSort

列に現在適用されているソート順序を表す文字列を取得します。 有効な値は、昇順の場合は'+'、降順の場合は'-'、列がソートされていない場合はnullです。

型 **string**

dataMap

生の値から列の表示値への変換に使用される**DataMap** を取得または設定します。

dataMap が関連付けられた列には、値をすばやく編集するためのドロップダウンボタンが表示されます。ドロップダウンボタンが表示されないようにするには、列の**showDropDown** プロパティをfalseに設定します。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

型 **DataMap**

dataType

列に格納される値の型を取得または設定します。

グリッドを編集するとき、値は適切な型に型変換されます。

型 **DataType**

describedById

列の説明を含む要素のIDを取得または設定します。

このIDは、列ヘッダ要素の **aria-describedby** 属性の値として使用されます。

型 **string**

dropDownCssClass

この列のドロップダウンに追加するCSSクラス名を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

型 **string**

format

未加工の値を列の表示値に変換するために使用される書式文字列を取得または設定します （**Globalize** を参照）。

型 **string**

grid

行または列を所有する**FlexGrid** を取得します。

継承元 **RowCol**
型 **FlexGrid**

header

列ヘッダに表示されるテキストを取得または設定します。

型 **string**

index

行または列の親コレクション内でのインデックスを取得します。

継承元 **RowCol**
型 **number**

inputType

この列の値の編集に使用されるHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは数値型の列では"tel"、その他のブール型以外の列型では"text"に設定されます。"tel"入力タイプを使用した場合、モバイルデバイスではマイナス記号や小数点を含む数値キーボードが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"に設定するか、単に"text"にしてみてください。

型 **string**

isContentHtml

この行または列にあるセルがプレーンテキストではなくHTMLコンテンツを含むかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

isReadOnly

行または列のセルを編集できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

isRequired

列の値が必須かどうかを決定する値を取得または設定します。

デフォルトでは、このプロパティはnullに設定されます。その場合、値は必須ですが、 非マスク文字列の列に空の文字列を含めることができます。

trueに設定した場合、値は必須で、空の文字列は許可されません。

falseに設定した場合は、null値と空の文字列が許可されます。

型 **boolean**

isSelected

行または列が選択されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isVisible

行または列が表示可能で、なおかつ折りたたまれていないかどうかを示す値を取得または設定します。

このプロパティは読み取り専用です。行または列の表示/非表示設定を変更するには、代わりに**visible** プロパティを使用してください。

継承元 **RowCol**
型 **boolean**

- mask

この列の値の編集時に使用するマスクを取得または設定します。

マスクの書式は**InputMask** コントロールで使用される書式と同じです。

指定する場合、このマスクは**format** プロパティの値と互換性がある必要があります。たとえば、'MM/dd/yyyy'と書式設定された 日付の入力にマスク'99/99/9999'を使用できます。

型 **string**

- maxLength

セルに入力できる最大の項目数を取得または設定します。

このプロパティをnullに設定すると、任意の数の文字を入力できます。

型 **number**

- maxWidth

列の最大幅を取得または設定します。

型 **number**

- minWidth

列の最小幅を取得または設定します。

型 **number**

- multiLine

この行または列にあるセルの内容が改行文字(\n).でラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- name

列の名前を取得または設定します。

列名を使用して、**getColumn** メソッドで列を取得できます。

型 **string**

pos

行または列の位置を取得します。

継承元 **RowCol**
型 **number**

quickAutoSize

この列のサイズを自動変更するときにグリッドが精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、この列の自動リサイズ処理が無効になります。 これをtrueに設定すると、グリッドの**quickAutoSize** プロパティの値にしたがって、この機能が有効になります。 null（デフォルト値）に設定すると、プレーンテキストを表示するテンプレートを持たない列の機能が有効になります。

型 **boolean**

renderSize

行または列のレンダリングサイズを取得します。 表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮したサイズが返されます。

継承元 **RowCol**
型 **number**

renderWidth

列のレンダリング幅を取得します。

列の表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮した幅が返されます。

型 **number**

showDropDown

グリッドで、この列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

型 **boolean**

size

行または列のサイズを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

継承元 **RowCol**
型 **number**

- sortMemberPath

この列をソートするときに使用するプロパティの名前を取得または設定します。

このプロパティは、**binding** プロパティによって指定されている値以外の値に基づいてソートを実行する場合に使用します。

このプロパティをnullに設定すると、**binding** プロパティの値を使用して列がソートされます。

型 **string**

- visible

行または列が表示されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- visibleIndex

非表示にした要素(**isVisible**)を無視し、親コレクション内の行または列のインデックスを取得します。

継承元 **RowCol**
型 **number**

- width

列の幅を取得または設定します。

列の幅は、正の数値（列幅のピクセル数を設定）、nullまたは負の数値（コレクションのデフォルトの列幅を使用）、または'{number}*'形式の文字列（スターサイズ指定）にすることができます。

スターサイズ指定オプションを使用すると、星の前の数字に比例して列の幅が決定されるXAMLスタイルの動的なサイズ設定が実行されます。たとえば、グリッドに3つの列があり、それぞれの幅が"100"、"*"、"3*"に設定されている場合、最初の列の幅は100ピクセルになり、2列目は残りスペースの1/4、3列目は残りスペースの3/4を占めます。

スターサイズ指定を使用すると、使用可能な幅いっぱいに自動的に広がる列を定義できます。たとえば、最後の列の幅を"*"に設定すると、最後の列がグリッドの幅いっぱいに拡張されて空スペースがなくなります。また、列の **minWidth** プロパティを設定して列の幅が狭くなりすぎないようにすることもできます。

型 **any**

- wordWrap

この行または列のセルの内容を使用可能な列幅に収まれるようにラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

メソッド

getAlignment

```
getAlignment(): string
```

列の実際の配置を取得します。

align プロパティがnullでない場合はその値が返されます。それ以外の場合は、列の**dataType** に基づいて配置が選択されます。

戻り値 **string**

getIsRequired

```
getIsRequired(): boolean
```

列が必須かどうかを判定する値を取得します。

isRequired プロパティがnullでない場合は、その値が返されます。 そうでない場合は、列の**dataType** に基づいて必須状態が判定されます。

デフォルトでは、文字列の列は、**dataMap** または **mask** 値が設定されていない限り 必須ではありません。他のすべてのデータ型は必須です。

戻り値 **boolean**

onPropertyChanged

```
onPropertyChanged(): void
```

オーナーリストをダーティとしてマークし、オーナーグリッドを更新します。

継承元 **RowCol**
戻り値 **void**

ColumnCollection クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
基本クラス **RowColCollection**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid コントロール内の**Column** オブジェクトのコレクションを表します。

コンストラクタ

- constructor

プロパティ

- defaultSize
- describedById
- firstVisibleIndex
- frozen
- isUpdating
- maxSize
- minSize
- visibleLength

メソッド

- beginUpdate
- canMoveElement
- clear
- deferUpdate
- endUpdate
- getColumn
- getItemAt
- getNextCell
- getTotalSize
- implementsInterface
- indexOf
- insert
- isFrozen
- moveElement
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

イベント

- collectionChanged

コンストラクタ

constructor

```
constructor(g: FlexGrid, defaultSize: number): RowColCollection
```

RowColCollection クラスの新しいインスタンスを初期化します。

パラメーター

- **g**: **FlexGrid**
  コレクションを所有する**FlexGrid** 。
- **defaultSize**: **number**
  コレクションの要素のデフォルトサイズ。

継承元 **RowColCollection**
戻り値 **RowColCollection**

プロパティ

defaultSize

コレクションの要素のデフォルトサイズを取得または設定します。

継承元 **RowColCollection**
型 **number**

describedById

列ヘッダの説明を含む要素のIDを取得または設定します。

このIDは、すべての列ヘッダ要素の **aria-describedby** 属性の値として使用されます。列固有の場合は、代わりに列の**describedById** を使用します。

型 **string**

firstVisibleIndex

最初の表示可能な列（アウトラインツリーが表示される列）のインデックスを取得します。

型

frozen

コレクションに含まれる静止行または静止列の数を取得または設定します。

静止行および静止列はスクロールされず、グリッドの上または左に（固定セルに隣接して）固定されます。ただし、固定セルとは異なり、静止セルは通常のセルと同じように選択して編集できます。

継承元 **RowColCollection**
型 **number**

isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

継承元 **ObservableArray**
型

maxSize

コレクションの要素の最大サイズを取得または設定します。

継承元 **RowColCollection**
型 **number**

minSize

コレクションの要素の最小サイズを取得または設定します。

継承元 **RowColCollection**
型 **number**

visibleLength

コレクション内(**isVisible**)に表示される要素の数を取得します。

継承元 **RowColCollection**
型 **number**

メソッド

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate**が呼び出されるまで通知を中断します。

継承元 **RowColCollection**
戻り値 **void**

canMoveElement

```
canMoveElement(src: number, dst: number): boolean
```

要素をある位置から別の位置に移動できるかどうかをチェックします。

パラメーター

- **src**: **number**
  移動する要素のインデックス。

- **dst**: **number**
  要素の移動先の位置。末尾に移動する場合は-1を指定します。

継承元 **RowColCollection**
戻り値 **boolean**

clear

```
clear(): void
```

配列からすべての項目を削除します。

継承元 **ObservableArray**
戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

継承元 **ObservableArray**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **ObservableArray**
戻り値 **void**

getColumn

```
getColumn(name: string): Column
```

名前または連結に基づいて列を取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **string**
  検索する名前または連結。

戻り値 **Column**

getItemAt

```
getItemAt(position: number): number
```

指定した物理位置にある要素のインデックスを取得します。

パラメーター

- **position**: **number**
  コレクションの項目の位置（ピクセル単位）。

継承元 **RowColCollection**
戻り値 **number**

getNextCell

```
getNextCell(index: number, move: SelMove, pageSize: number): void
```

選択を変更するために次の表示可能なセルを検索します。

パラメーター

- **index**: **number**
  検索の開始インデックス。

- **move**: **SelMove**
  移動のタイプ（サイズと方向）。

- **pageSize**: **number**
  ページのサイズ（移動がページアップ/ダウンの場合）。

継承元 **RowColCollection**
戻り値 **void**

getTotalSize

```
getTotalSize(): number
```

コレクションの要素の合計サイズを取得します。

継承元 **RowColCollection**
戻り値 **number**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **ObservableArray**
戻り値 **boolean**

indexOf

```
indexOf(name: any): number
```

名前または連結に基づいて列のインデックスを取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **any**
  検索する名前または連結。

戻り値 **number**

insert

```
insert(index: number, item: any): void
```

配列内の指定した位置に項目を挿入します。

パラメーター

- **index**: **number**
  項目を追加する位置。

- **item**: **any**
  配列に追加する項目。

継承元 **ObservableArray**
戻り値 **void**

isFrozen

```
isFrozen(index: number): boolean
```

行または列が静止行または静止列かどうかをチェックします。

パラメーター

- **index**: **number**
  チェックする行または列のインデックス。

継承元 **RowColCollection**
戻り値 **boolean**

moveElement

```
moveElement(src: number, dst: number): void
```

要素をある位置から別の位置に移動します。

パラメーター

- **src**: **number**
  移動する要素のインデックス。

- **dst**: **number**
  要素の移動先を示す位置（末尾に移動する場合は-1）。

継承元 **RowColCollection**
戻り値 **void**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

ダーティ状態を追跡し、変更時にグリッドを無効化します。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL

継承元 **RowColCollection**
戻り値 **void**

push

```
push(item: any): number
```

配列の最後に項目を追加します。

パラメーター

- **item**: **any**
  配列に追加する項目。

継承元 **RowColCollection**
戻り値 **number**

remove

```
remove(item: any): boolean
```

配列から項目を削除します。

パラメーター

- **item**: **any**
  削除する項目。

継承元 **ObservableArray**
戻り値 **boolean**

removeAt

```
removeAt(index: number): void
```

配列内の指定した位置にある項目を削除します。

パラメーター

- **index**: **number**
  削除する項目の位置。

継承元 **ObservableArray**
戻り値 **void**

setAt

```
setAt(index: number, item: any): void
```

配列内の指定した位置にある項目を設定します。

パラメーター

- **index**: **number**
  項目を設定する位置。

- **item**: **any**
  配列に設定する項目。

継承元 **ObservableArray**
戻り値 **void**

slice

```
slice(begin?: number, end?: number): any[]
```

配列の一部のシャローコピーを作成します。

パラメーター

- **begin**: **number** OPTIONAL
  コピーを開始する位置。

- **end**: **number** OPTIONAL
  コピーを終了する位置。

継承元 **ObservableArray**
戻り値 **any[]**

sort

```
sort(compareFn?: Function): this
```

配列の要素を適切な位置にソートします。

パラメーター

- **compareFn**: **Function** OPTIONAL
  ソート順序を定義する関数。 この関数は2つの引数を受け取り、-1（最初の引数の方が2番目の引数より小さい場合）、+1（大きい場合）、0（等しい場合）のいずれかの値を返す必要があります。 これを省略した場合は、各要素の文字列変換に従って辞書順にソートされます。

継承元 **ObservableArray**
戻り値 **this**

splice

```
splice(index: number, count: number, item?: any): any[]
```

配列からの項目の削除、または配列への項目の追加を行います。

パラメーター

- **index**: **number**
  項目を追加または削除する位置。

- **count**: **number**
  配列から削除する項目の数。

- **item**: **any** OPTIONAL
  配列に追加する項目。

継承元 **RowColCollection**
戻り値 **any[]**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 **ObservableArray**
引数 **NotifyCollectionChangedEventArgs**

DataMap クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**

列の**dataMap** プロパティで使用するデータマップを表します。

データマップは、グリッドに自動検索機能を提供します。たとえば、顧客のIDの代わりに顧客名、RGB値の代わりに色名を 表示できます。

以下のコードは、グリッドを製品のコレクションに連結してから、**DataMap** をグリッドの'CategoryID'列に割り当てて、グリッドにID自体ではなくカテゴリ名を表示します。

グリッドは、編集にもデータマップを利用します。**wijmo.input**モジュールをロードした場合は、データマップされた列の編集時に、マップの値を含むドロップダウンリストが表示されます。

```
// グリッドを製品に結合しますvar flex = new wijmo.grid.FlexGrid();flex.itemsSource = products;// IDの代わりにカテゴリ名を表示するようにCategoryID列をマップしますvar col = flex.columns.getColumn('CategoryID');col.dataMap = new wijmo.grid.DataMap(categories, 'CategoryID', 'CategoryName');
```

一般に、データマップは列全体に適用されます。ただし、別の列の値に基づいて、セルに使用するオプションを制限することもできます。たとえば、"Country"列と"City"列がある場合は、現在の国に基づいて都市を制限することがよくあります。

このような「動的」データマップを実装する方法は、次の2つあります。

1. **DataMap** が単なる文字列のリストである場合は、グリッドが編集モードになる 前に変更を加えます。この場合、セルには表示される文字列が含まれ、 マップを変更しても同じ列の他のセルには影響しません。 次のサンプルに例を示します。 show me (http://jsfiddle.net/Wijmo5/8brL80r8/)。
2. **DataMap** が実際のマップの場合（キー値をセル内に格納し、対応する文字列を表示する）は、 ドロップダウンに表示する値を制限するためのフィルタを適用します。**DataMap** には同じキーと値が格納されたままなので、 同じ列の他のセルはフィルタの影響を受けません。 次のサンプルに例を示します。 show me (http://jsfiddle.net/Wijmo5/xborLd4t/)。

場合によっては、列挙を表す**DataMap** を作成することができます。 上記を実現するには、次のコードを使用できます。

```
// 指定された列挙型のDataMapを構築します
 function getDataMap(enumClass) {
     var pairs = [];
     for (var key in enumClass) {
         var val = parseInt(key);
         if (!isNaN(val)) {
             pairs.push({ key: val, name: enumClass[val] });
         }
     }
     return new wijmo.grid.DataMap(pairs, 'key', 'name');
}
```

コンストラクタ

- constructor

プロパティ

- collectionView
- displayMemberPath
- isEditable
- selectedValuePath
- sortByDisplayValues

メソッド

- getDisplayValue
- getDisplayValues
- getKeyValue
- getKeyValues
- onMapChanged

イベント

- mapChanged

コンストラクタ

constructor

```
constructor(itemsSource: any, selectedValuePath?: string, displayMemberPath?: string): DataMap
```

DataMap クラスの新しいインスタンスを初期化します。

パラメーター

- **itemsSource**: **any**
  マップする項目を含む配列または**ICollectionView** 。
- **selectedValuePath**: **string** OPTIONAL
  キー（データ値）を含むプロパティの名前。
- **displayMemberPath**: **string** OPTIONAL
  項目のビジュアル表現として使用するプロパティ名。

戻り値 **DataMap**

プロパティ

- collectionView

マップデータを含む**ICollectionView** オブジェクトを取得します。

型 **ICollectionView**

displayMemberPath

項目のビジュアル表現として使用するプロパティ名を取得します。

型 **string**

isEditable

ユーザーが**DataMap** に存在しない値を入力できるかどうかを示す値を取得または設定します。

DataMap を編集可能にするには、**selectedValuePath** と**displayMemberPath** が同じ値に設定されている必要があります。

型 **boolean**

selectedValuePath

項目のキー（データ値）として使用するプロパティ名を取得します。

型 **string**

sortByDisplayValues

データのソート時に、マッピングされた（表示）値と生の値のどちらを使用するかを決定する値を取得または設定します。

型 **boolean**

メソッド

getDisplayValue

```
getDisplayValue(key: any): any
```

指定されたキーに対応する表示値を取得します。

パラメーター

- **key**: **any**
  得する項目のキー。

戻り値 **any**

getDisplayValues

```
getDisplayValues(dataItem?: any): string[]
```

マップのすべての表示値を含む配列を取得します。

パラメーター

- **dataItem**: **any** OPTIONAL
  取得する表示項目に対応するデータ項目。 このパラメータはオプションです。 指定しない場合は、可能なすべての表示値が返されます。

戻り値 **string[]**

getKeyValue

```
getKeyValue(displayValue: string): any
```

指定された表示値に対応するキーを取得します。

パラメーター

- **displayValue**: **string**
  取得する項目の表示値。

戻り値 **any**

getKeyValues

```
getKeyValues(): string[]
```

マップのすべてのキーを含む配列を取得します。

戻り値 **string[]**

onMapChanged

```
onMapChanged(e?: EventArgs): void
```

mapChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

イベント

mapChanged

マップデータが変更されたときに発生します。

引数 **EventArgs**

FlexGrid クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
基本クラス **Control**
派生クラス **MultiRow**, **FlexSheet**, **PivotGrid**, **WjFlexGrid**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid コントロールは、データを表形式で表示・編集する強力かつ柔軟な手段を提供します。

FlexGrid コントロールは機能豊富なグリッドであり、複数の選択モード、ソート、列の順序変更、グループ化、フィルタリング、編集、カスタムセル、XAMLスタイルのスターサイズ指定、行および列の仮想化といったグリッドによく見られる機能をすべて備えています。

FlexGrid コントロールは、次のキーボードコマンドをサポートしています。

キーの組み合わせ	アクション
Shift + Space	行全体を選択する
Control + Space	列全体を選択する
F2	アクティブなセルを編集します
Space	編集の開始またはチェックボックスの切り替え
Control + A	グリッドの内容全体を選択します
←/→	現在の選択範囲の左/右にあるセルを選択するか、またはグループ行を折りたたむ/展開します
Shift + ←/→	選択の左/右にあるセルを選択範囲に追加するように選択範囲を拡大します
↑/↓	選択範囲の１つ上または下にあるセルを選択します
Shift + ↑/↓	選択範囲の上または下にあるセルを選択範囲に追加するように選択範囲を拡大します
Alt + ↑/↓	現在のセルのリストボックスエディタをドロップダウン表示します
PageUp/Down	選択範囲の1ページ上または下のセルを選択します
Shift + PageUp/Down	選択範囲に1ページ上のセルまたは1ページ下のセルを含めるように選択範囲を拡張します
Alt + PageUp/Down	選択範囲を最初または最後の行に移動します
Shift + Alt + PageUp/Down	最初または最後の行を含むように選択範囲を拡張します
Home/End	選択範囲を最初または最後の列に移動します
Shift + Home/End	選択範囲を最初または最後の列に移動します
Ctrl + Home/End	選択範囲を最初または最後の行と列に移動します
Shift + Ctrl + Home/End	最初または最後の行と列を含むように選択範囲を拡張します
Escape	現在のセルまたは行の編集操作をキャンセルします
Tab	選択範囲をページ上の次のフォーカス可能な要素に移動します（デフォルトでは、**keyActionTab** プロパティを使用してオーバーライドできます）
Enter	編集モードを終了し、選択範囲を現在のセルの下のセルに移動します（デフォルトでは、**keyActionEnter** プロパティを使用してオーバーライドできます）
Delete, Backspace	**allowDelete** プロパティがtrueに設定されている場合現在選択されている行を削除します。または、値が不要な場合選択したセルの内容を消去します。
Control + C or Control + Insert	選択範囲をクリップボードにコピーします（**autoClipboard** プロパティがtrueに設定されている場合）
Control + V or Shift + Insert	選択した領域にクリップボードの内容を貼り付けます（**autoClipboard** プロパティがtrueに設定されている場合）

サンプル

Show me (https://jsfiddle.net/Wijmo5/6GB66)

コンストラクタ

- constructor

プロパティ

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting

- autoClipboard
- autoGenerateColumns
- autoScroll
- autoSearch
- autoSizeMode
- bottomLeftCells
- cellFactory
- cells
- childItemsPath
- clientSize
- cloneFrozenCells
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- deferResizing
- editableCollectionView
- editRange
- frozenColumns
- frozenRows
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemValidator
- keyActionEnter
- keyActionTab
- mergeManager
- newRowAtTop
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- rightToLeft
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollSize
- selectedItems
- selectedRows

- selection
- selectionMode
- showAlternatingRows
- showDropDown
- showErrors
- showGroups
- showMarquee
- showSelectedHeaders
- showSort
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- validateEdits
- viewRange
- virtualizationThreshold

メソッド

- addEventListener
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseGroupsToLevel
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow

- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onItemsSourceChanging
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onRefreshed
- onRefreshing
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView

- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- startEditing
- toggleDropDownList

イベント

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- itemsSourceChanging
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- refreshed
- refreshing
- resizedColumn
- resizedRow
- resizingColumn

- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged
- selectionChanging
- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout
- updatingView

コンストラクタ

constructor

```
constructor(element: any, options?): FlexGrid
```

FlexGrid クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **FlexGrid**

プロパティ

activeEditor

現在アクティブになっているセルエディタを表す**HTMLInputElement**を取得します。

型 **HTMLInputElement**

allowAddNew

ユーザーがソースコレクションに項目を追加できるようにグリッドに新規行テンプレートを表示するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、新規行テンプレートは表示されません。

型 **boolean**

allowDelete

ユーザーが［Delete］キーを押したときにグリッドの選択されている行を削除するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、選択されている行は削除されません。

型 **boolean**

allowDragging

ユーザーがマウスを使用して行、列、またはその両方をドラッグできるかどうかを決定する値を取得または設定します。

autoScroll プロパティがtrueに設定されている場合、ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされます。

グリッドでは、列のドラッグがデフォルトで有効になっています。

グリッドが連結モードでは、行をドラッグするには特別な 考慮が必要になります。

グリッドが連結モードでは、行をドラッグするとデータソースとの同期が失われます（たとえば行4は6行として参照されることがあります）。 これを回避するには、**draggedRow** イベントを処理し、データを新しい行の位置と同期させる必要があります。

また、**allowSorting** プロパティをfalseに設定する必要があります。そうしないと、行の順序がデータによって決定され、行をドラッグするのは無意味になります。

次のフィドルでは、連結グリッドでの行のドラッグを実行しています。 **Bound Row Dragging (http://jsfiddle.net/Wijmo5/kyg0qsda/)**.

型 **AllowDragging**

allowMerging

グリッドのどの部分のセル結合を許可するかを取得または設定します。

型 **AllowMerging**

allowResizing

ユーザーがマウスを使用して行、列、またはその両方をサイズ変更できるかどうかを決定する値を取得または設定します。

サイズ変更が可能な場合は、列ヘッダーセルの右端をドラッグして列を、行ヘッダーセルの下端をドラッグして行をサイズ変更できます。

ヘッダーセルの端をダブルクリックして、行および列をコンテンツに合わせて自動的にサイズ変更することもできます。 自動サイズ変更の動作は、**autoSizeMode** プロパティを使用してカスタマイズできます。

型 **AllowResizing**

allowSorting

ユーザーが列ヘッダーセルをクリックして列をソートできるかどうかを決定する値を取得または設定します。

型 **boolean**

autoClipboard

グリッドがクリップボードショートカットを処理するかどうかを決定する値を取得または設定します。

次のクリップボードショートカットがあります。

［Ctrl］+［C］、［Ctrl］+［Ins］	グリッドの選択範囲をクリップボードにコピーします。
［Ctrl］+［V］、［Shift］+［Ins］	クリップボードのテキストをグリッドの選択範囲に貼り付けます。

クリップボード操作は、表示されている行および列だけが対象となります。

読み取り専用のセルは、貼り付け操作の影響を受けません。

型 **boolean**

autoGenerateColumns

グリッドが**itemsSource** に基づいて列を自動的に生成するかどうかを 決定する値を取得または設定します。

列の生成は、少なくとも1項目を含む**itemsSource** プロパティに依存します。このデータ項目が検査され、列が作成されて、プリミティブ値（数値、文字列、Boolean、またはDate）を含むプロパティに連結されます。

プロパティをnullに設定すると、適切なタイプを推測する方法がないため、列は生成されません。このような場合は、**autoGenerateColumns** プロパティをfalseに設定し、明示的に列を作成する必要があります。次に例を示します。

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // データ項目にnull値が含まれる場合があります
  columns: [                  // そのため、列を明示的に定義します
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

型 **boolean**

autoScroll

ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされるかどうかを決定する値を取得または設定します。

行と列のドラッグは、**allowDragging** プロパティによって制御されます。

このプロパティはデフォルトでtrueに設定されます。

型 **boolean**

autoSearch

ユーザーが読み取り専用セルに入力するに伴ってグリッドでセルを検索するかどうかを決定する値を取得または設定します。

検索が現在選択されている列(編集不可能な場合)で行われます。検索は現在選択されている行から始まり、大文字と小文字を区別されません。

型 **boolean**

autoSizeMode

行または列のサイズを自動設定するときにどのセルを考慮に入れるかを取得または設定します。

このプロパティは、ユーザーが列ヘッダの端をダブルクリックしたときの動作を制御します。

デフォルトでは、その列のヘッダセルとデータセルの内容に基づいて列の幅が自動的に設定されます。このプロパティを使用すると、ヘッダのみまたはデータのみに基づいて列の幅を設定するように変更できます。

型 **AutoSizeMode**

bottomLeftCells

左下のセルを含む**GridPanel** を取得します。

bottomLeftCells パネルは、行ヘッダの下、**columnFooters** パネルの左に表示されます。

型 **GridPanel**

cellFactory

このグリッドのセルを作成および更新する**CellFactory** を取得または設定します。

型 **CellFactory**

cells

データセルを含む**GridPanel** を取得します。

型 **GridPanel**

childItemsPath

階層グリッドで子の行の生成に使用される 1 つ以上のプロパティの名前を取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `childItemsPath = 'items'` ）。

項目の異なるレベルに異なる名前を持つ子項目がある場合は、このプロパティに、各レベルの子項目を含むプロパティの名前から成る配列を設定します。（たとえば、 `childItemsPath = ['checks','earnings'];` ）。

サンプル

Show me (https://jsfiddle.net/Wijmo5/kk9j93bL)

型 **any**

clientSize

コントロールのクライアントサイズを取得します（コントロールのサイズからヘッダーとスクロールバーを引いた値）。

型 **Size**

cloneFrozenCells

スクロール中に知覚されるパフォーマンスを向上させるには、FlexGridがフリーズされたセルを複製し、別の要素に表示するかどうかを決定する値を取得または設定します。

このプロパティのデフォルト値はnullであり、ブラウザーによって一番適当な設定が選択されます。

型 **boolean**

collectionView

グリッドデータを含む**ICollectionView** を取得します。

型 **ICollectionView**

columnFooters

列フッターセルを保持する**GridPanel** を取得します。

columnFooters パネルは、グリッドセルの下、**bottomLeftCells** パネルの右に表示されます。これを使用して、グリッドデータの下にサマリー情報を表示できます。

以下の例では、**columnFooters** パネルに 1行を追加して、**aggregate** プロパティが設定された列に サマリーデータを表示する方法を示します。

```
function addFooterRow(flex) {

  // 集計を表示するGroupRowを作成します
  var row = new wijmo.grid.GroupRow();

  // 列フッターパネルに行を追加します
  flex.columnFooters.rows.push(row);

  // ヘッダーにシグマを表示します
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

型 **GridPanel**

columnHeaders

列ヘッダセルを含む**GridPanel** を取得します。

型 **GridPanel**

columnLayout

現在の列レイアウトを定義するJSON文字列を取得または設定します。

列レイアウト文字列は、列とそのプロパティを含む配列を表します。これを使用して、ユーザーが定義した列レイアウトをセッションを越えて保持できます。また、ユーザーが列レイアウトを変更できるアプリケーションで元に戻す/やり直し機能を実装することもできます。

データマップはシリアル化できないため、列レイアウト文字列に**dataMap**　プロパティは含まれていません。

型 **string**

columns

グリッドの列コレクションを取得します。

型 **ColumnCollection**

controlRect

コントロールの外接矩形（ページ座標単位）を取得します。

型 **Rect**

STATIC controlTemplate

FlexGrid コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

deferResizing

ユーザーがマウスボタンを放すまで、行および列のサイズ変更を遅らせるかどうかを決定する値を取得または設定します。

デフォルトでは、**deferResizing** はfalseに設定されており、ユーザーがマウスをドラッグするに伴って行および列がサイズ変更されます。このプロパティをtrueに設定すると、グリッドにサイズ変更マーカーが表示され、ユーザーがマウスボタンを放したときに初めて行または列のサイズが変更されます。

型 **boolean**

editableCollectionView

グリッドデータを含む**IEditableCollectionView** を取得します。

型 **IEditableCollectionView**

editRange

現在編集中のセルを識別する**CellRange** を取得します。

型 **CellRange**

frozenColumns

固定された列の数を取得または設定します。

固定された列は水平方向にスクロールできませんが、セルは選択および編集できます。

型 **number**

frozenRows

静止行の数を取得または設定します。

固定された行は垂直方向にスクロールできませんが、セルは選択および編集できます。

型 **number**

groupHeaderFormat

グループヘッダーコンテンツを作成するために使用される書式文字列を取得または設定します。

この文字列は、任意のテキストと次の置換文字列を含むことができます。

- **{name}**：グループ化されるプロパティの名前。
- **{value}**：グループ化されるプロパティの値。
- **{level}**：グループレベル。
- **{count}**：このグループ内にある項目の合計数。

列がグループ化プロパティに連結されている場合は、列ヘッダーを使用して パラメータを置換し、列の書式とデータマップを使用して `{value}` パラメータを計算します。 列がない場合は、代わりにグループ情報が使用されます。

グループプロパティに連結された非表示の列を追加して、グループヘッダーセルの書式設定をカスタマイズすることもできます。

このプロパティのデフォルト値は
`'{name}: <b>{value}</b>({count:n0} items)'` です。 これは、 `'Country: UK (12 items)'` や
`'Country: Japan (8 items)'` のようなグループヘッダーを作成します。

型 **string**

headersVisibility

行ヘッダおよび列ヘッダが表示されるかどうかを決定する値を取得または設定します。

型 **HeadersVisibility**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

imeEnabled

編集モードでないときに、グリッドがIME（Input Method Editor）をサポートするかどうかを決定する値を取得または設定します。

このプロパティは、日本語、中国語、韓国語など、IMEサポートを必要とする言語のサイト/アプリケーションにのみ関係します。

型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してセル値を変更できるかどうかを判定する値を取得または設定します。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

このグリッドのセルのカスタマイズに使用されるフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意のセルに任意の内容を追加できます。そのため、グリッドセルの外観と動作を非常に柔軟に制御することが可能です。

この関数は、セルを含む**GridPanel**、セルの行インデックスと列インデックス、セルを表すHTML要素の4つのパラメーターをとります。通常は、関数内でセル要素の**innerHTML** プロパティを変更するようにします。

例:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == wijmo.grid.CellType.Cell) {

    // セルにスパークラインを描画します。
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

FlexGridはセルをリサイクルするため、**itemFormatter** によってセルのスタイル属性を変更する場合は、 セルの適切でないスタイル属性を事前にリセットする必要があります。例:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // カスタマイズする属性をリセットします。
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // このセルのcolorおよびbackgroundColor属性をカスタマイズします。
  ...
}
```

複数のクライアントがグリッドのレンダリングをカスタマイズできるようにする場合は（たとえば、ディレクティブや再利用可能なライブラリを作成するときなど）、代わりに**formatItem** イベントを使用することを検討してください。このイベントを使用すると、複数のクライアントがそれぞれ独自のハンドラをアタッチできます。

型 **Function**

itemsSource

グリッドに表示される項目を含む配列または**ICollectionView** を取得または設定します。

型 **any**

itemValidator

セルに有効なデータが含まれているかどうかを決定するバリデータ関数を取得または設定します。

指定された場合、バリデータ関数はセルの行インデックスと列インデックスを含む2つのパラメータをとり、エラーの説明を含む文字列を返す必要があります。

このプロパティは、バインドされていないグリッドを処理する場合に特に便利です。バインドされたグリッドは、代わりに**getError** プロパティを使用して検証できます。

この例では、セルのすぐ上のセルと同じデータがセルに含まれないようにする方法を示します。

```
// 上のセルは、現在のセルと同じ値が含まれていないことを確認します
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'これは重複した値です...'
    }
  }
  return null; // エラーなし
}
```

型 **Function**

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は**MoveDown** となり、コントロールがカーソルを次の行に移動します。 この動作は標準的なExcel式の動作です。

型 **KeyAction**

keyActionTab

［Tab］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は、**None** となります。これにより、Tabキーが押されたときにブラウザ上で次または前のコントロールを選択できます。これは、ページのアクセシビリティを向上させるために推奨される設定になります。

以前のバージョンでは、デフォルトは**Cycle** に設定されていました。これにより、コントロールがカーソルを、グリッドを横切って下部に移動しました。これは標準的なExcelの動作ですが、アクセシビリティには適していません。

また、**CycleOut** の設定が存在します。これにより、カーソルがセルを通じて移動（**Cycle** ）してから、最後または最初のセルが選択されたときにページの次/前のコントロールに移動します 。

型 **KeyAction**

mergeManager

セルの結合方法を決定する**MergeManager** オブジェクトを取得または設定します。

型 **MergeManager**

newRowAtTop

新しい行テンプレートをグリッドの上部に配置するか、下部に配置するかを示す値を取得または設定します。

newRowAtTop プロパティをtrueに設定した場合、新しい行テンプレートを常に表示したままにする場合は、**frozenRows** プロパティを 1に設定します。これにより、新しい行テンプレートは上部に固定され、ビューからスクロールオフされなくなります。

allowAddNew プロパティがtrueに設定されている場合、および**itemsSource** オブジェクトが新しい項目の追加をサポートしている場合にのみ、新しい行テンプレートが表示されます。

型 **boolean**

preserveOutlineState

データがリフレッシュされたときに、グリッドがノードの展開/折りたたみ状態を保持するかどうかを決定する値を取得または設定します。

preserveOutlineState プロパティの実装は、JavaScriptの**Map** オブジェクトに基づいています。このオブジェクトはIE 9およびIE 10で利用できません。

型 **boolean**

preserveSelectedState

データがリフレッシュされたときに、グリッドが行の選択状態を保持するかどうかを決定する値を取得または設定します。

型 **boolean**

quickAutoSize

グリッドが列のサイズを自動調整するときに精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、自動サイズ変更の機能が無効になります。このプロパティをtrueに設定すると、各列の**quickAutoSize** プロパティの値に従って、その機能が有効になります。null（デフォルト値）に設定した場合、カスタムの**itemFormatter** を持たないグリッドの機能、または**itemFormatter** イベントにアタッチされたハンドラの機能が有効になります。

型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rowHeaderPath

行ヘッダーセルを作成するために使用されるプロパティの名前を取得または設定します。

行ヘッダーセルは表示されないし、選択することもできません。アクセシビリティツールと組み合わせて使用することを意図しています。

型 **string**

rowHeaders

行ヘッダセルを含む**GridPanel** を取得します。

型 **GridPanel**

- rows

グリッドの行コレクションを取得します。

型 **RowCollection**

- scrollPosition

グリッドのスクロールバーの値を表す**Point** を取得または設定します。

型 **Point**

- scrollSize

グリッド内容のサイズ（ピクセル単位）を取得します。

型 **Size**

- selectedItems

現在選択されているデータ項目を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

型 **any**[]

- selectedRows

現在選択されている行を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

型 **any**[]

- selection

現在の選択を取得または設定します。

型 **CellRange**

- selectionMode

現在の選択モードを取得または設定します。

型 **SelectionMode**

- showAlternatingRows

グリッドで交互表示行のセルに'wj-alt'クラスを追加するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、CSSを変更しなくても、交互表示行スタイルを無効にできます。

型 **boolean**

- showDropDown

グリッドで、**showDropDown** プロパティがtrueに設定されている列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

型 **boolean**

- showErrors

グリッドで、検証エラーがあるセルとエラーの説明を含むツールチップに'wj-state-invalid' クラスを追加するかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**itemValidator** または**getError** プロパティを使用して、検証エラーを検出します。

型 **boolean**

- showGroups

データグループを区切るためにグリッドにグループ行を挿入するかどうかを決定する値を 取得または設定します。

データグループを作成するには、グリッドの **itemsSource** として使用される**ICollectionView** オブジェクトの**groupDescriptions** プロパティを変更します。

型 **boolean**

- showMarquee

現在の選択範囲の周囲にマーキー要素を表示するかどうかを示す値を取得または設定します。

型 **boolean**

- showSelectedHeaders

選択されているヘッダセルを示すためにクラス名を追加するかどうかを示す値を取得または設定します。

型 **HeadersVisibility**

- showSort

グリッドで、列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

ソートは、グリッドの**itemsSource** として使用される**ICollectionView** オブジェクトの**sortDescriptions** プロパティによって制御されます。

型 **boolean**

- sortRowIndex

ソートインジケーターを表示する列ヘッダパネルの行のインデックスを取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、 **columnHeaders** パネルの最後の行がソート行になります。

型 **number**

stickyHeaders

ユーザーがドキュメントをスクロールしたときに、列ヘッダーを表示したままにするかどうかを決定する値を取得または設定します。

型 **boolean**

topLeftCells

左上のセル（列ヘッダーの左）を含む**GridPanel** を取得します。

型 **GridPanel**

treeIndent

異なるレベルの行グループをオフセットするインデントを取得または設定します。

型 **number**

validateEdits

検証に失敗した編集をユーザーがコミットしようとしたときに、グリッドを編集モードのままにするかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**getError** メソッドを呼び出して、検証エラーを検出します。

型 **boolean**

viewRange

現在表示されているセルの範囲を取得します。

型 **CellRange**

virtualizationThreshold

仮想化を有効にするために必要な最小行数、最小列数、またはその両方を取得または設定します。

このプロパティはデフォルトでゼロに設定されます。つまり、仮想化は常に有効ということです。これにより、バインディングパフォーマンスとメモリー必要量が向上しますが、スクロール時のパフォーマンス低下は犠牲になります。

グリッドの行数が少ない場合（約50〜100）、このプロパティを150などのやや高い値に設定することで、スクロールのパフォーマンスを向上させることができます。これにより、仮想化が無効になり連結処理が遅くなりますが、認識されるスクロールのパフォーマンスが向上する可能性があります。たとえば、以下のコードは、データソースに150を超える項目がある場合、グリッドがセルを仮想化します。

```
// 150を超える項目がある場合はグリッドを仮想化します
theGrid.virtualizationThreshold = 150;
```

このプロパティを200より大きい値に設定することは推奨されません。ロード処理の時間が長くなり、グリッドはすべての行のセルを作成しているときに数秒間フリーズするため、ページ上の要素数が多いためブラウザが遅くなります。

行と列に別々の仮想化しきい値を設定する場合は、**virtualizationThreshold** プロパティを2つの数値を含む配列に設定できます。この場合、最初の数値は行のしきい値として使用され、2番目の数値は列のしきい値として使用されます。たとえば、次のコードでは、グリッドは行を仮想化しますが、列を仮想化しません。

```
// 行（しきい値0）は仮想化しますが、列は仮想化しません（しきい値10,000）
theGrid.virtualizationThreshold = [0, 10000];
```

型 **any**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 `Control`
戻り値 **HTMLElement**

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

列をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **c**: **number**
  サイズ変更する列のインデックス。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

戻り値 **void**

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

列の範囲をその内容がちょうど収まるようにサイズ変更します。

測定される行の範囲は常に、現在表示されているすべての行 + 現在表示されていない最大2,000行です。グリッドに大量のデータが含まれている場合には（たとえば、50,000行など）、すべての行を測定すると非常に時間がかかる可能性があるため、すべての行は測定されません。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstColumn**: **number** OPTIONAL
  サイズ変更する最初の列のインデックス（デフォルトは最初の列）。
- **lastColumn**: **number** OPTIONAL
  サイズ変更する最後の列のインデックス（デフォルトは最後の列）。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

戻り値 **void**

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

行をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **r**: **number**
  サイズ変更する行のインデックス。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の列であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

戻り値 **void**

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

行の範囲をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックス。
- **lastRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックスす。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の行であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

戻り値 **void**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

canEditCell

```
canEditCell(r: number, c: number): void
```

指定されたセルが編集可能かどうかを示す値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

戻り値 **void**

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

すべてのグループ行を指定したレベルに折りたたみます。

パラメーター

- **level**: **number**
  表示する最大のグループレベル。

戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

戻り値 **boolean**

focus

```
focus(): void
```

グリッド全体をスクロールしてビューに入れることなくグリッドにフォーカスを設定するようにオーバーライドされます。

戻り値 **void**

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セル要素の範囲（ビューポート座標単位）を取得します。

このメソッドは、**cells** パネル内のセル（スクロール可能なデータセル）の範囲を返します。その他のパネルにあるセルの範囲を取得するには、該当する**GridPanel** オブジェクトの**getCellBoundingRect** メソッドを使用してください。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

戻り値 **Rect**

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

グリッドのスクロール可能領域内のセルに格納された値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

戻り値 **any**

getClipString

```
getClipString(rng?: CellRange): string
```

CellRange の内容を、クリップボードへのコピーに適した文字列として取得します。

非表示の行および列はクリップボード文字列に含まれません。

パラメーター

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

戻り値 **string**

getColumn

```
getColumn(name: string): Column
```

名前または連結に基づいて列を取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **string**
  検索する名前または連結。

戻り値 **Column**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

GridPanel 内のセルの結合範囲を示す**CellRange** を取得します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。
- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **clip**: **boolean** OPTIONAL
  結合範囲をグリッドの現在のビュー範囲にクリップするかどうか。

戻り値 **CellRange**

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

パラメーター

- **r**: **number**
  検査するセルの行インデックス。

- **c**: **number**
  検査するセルの列インデックス。

戻り値 **SelectedState**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

指定されたポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

次に例を示します。

```
// ユーザーがグリッドをクリックしたときに、ポイントをヒットテストします
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

パラメーター

- **pt**: **any**
  調べる**Point** （ページ座標単位）、MouseEventオブジェクト、またはポイントのX座標。

- **y**: **any** OPTIONAL
  ポイントのY座標（ページ座標単位、最初のパラメーターが数値の場合）。

戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

指定したCellRangeがこのグリッドの行および列コレクションに対して有効かどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックする範囲。

戻り値 **boolean**

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

autoSizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

autoSizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

autoSizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

autoSizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

beginningEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

cellEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

cellEditEnding イベントを発生させます。

パラメーター

- **e**: **CellEditEndingEventArgs**
  イベントデータを含む**CellEditEndingEventArgs** 。

戻り値 **boolean**

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

copied イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

copying イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

deletedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

deletingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

draggedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

draggedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

draggingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

draggingColumnOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

draggingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

draggingRowOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

groupCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

groupCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onItemsSourceChanging

```
onItemsSourceChanging(e: CancelEventArgs): boolean
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

戻り値 **boolean**

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

loadedRows イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

loadingRows イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

pasted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

pastedCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

pasting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

pastingCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

prepareCellForEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

resizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

resizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

resizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

resizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

rowAdded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

rowEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

rowEditEnding イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

rowEditStarted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

rowEditStarting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

scrollPositionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

selectionChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

sortedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **void**

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

sortingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

updatedLayout イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

updatingLayout イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

戻り値 **boolean**

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

updatingView イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

戻り値 **boolean**

refresh

```
refresh(fullUpdate?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean**
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

- **recycle**: **boolean** OPTIONAL
  既存の要素を再利用するかどうか。

- **state**: **boolean** OPTIONAL
  既存の要素を保持してその状態を更新するかどうか。

戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

scrollIntoView

```
scrollIntoView(r: number, c: number, refresh?: boolean): boolean
```

指定したセルが画面に入るようにグリッドをスクロールします。

負の行と列のインデックスは無視されるので、以下を呼び出すと、

```
grid.scrollIntoView(200, -1);
```

グリッドは200行目を表示するように垂直にスクロールしますが、水平にスクロールしません。

パラメーター

- **r**: **number**
  画面に入るようにスクロールする行のインデックス
- **c**: **number**
  画面に入るようにスクロールする列のインデックス。
- **refresh**: **boolean** OPTIONAL
  スクロールした新しい位置をすぐ表示するためにグリッドを更新する必要があるかどうかを決定するオプションのパラメータ。

戻り値 **boolean**

select

```
select(rng: any, show?: any): void
```

セル範囲を選択し、オプションでそのセル範囲が画面に入るようにスクロールします。

select メソッドは、**CellRange** 、と新しい選択範囲を画面に入るようにスクロールするかどうかを示すオプションのブール型パラメータを渡すことで呼び出すことができます。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(new CellRange(1, 1), true);

// 選択範囲（1,1）~（2,4）を指定し、画面に入るようにスクロールしません。
grid.select(new CellRange(1, 1, 2, 4), false);
```

select メソッドを呼び出してインデックスまたは選択する行と列を渡すこともできます。この場合、選択範囲が画面に入るようにスクロールします。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(1, 1);
```

パラメーター

- **rng**: **any**
  選択する範囲。
- **show**: **any** OPTIONAL
  新しい選択範囲が画面に入るようにスクロールするかどうか。

戻り値 **void**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

グリッドのスクロール可能領域内のセルの値を設定します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **value**: **any**
  セルに格納する値。
- **coerce**: **boolean** OPTIONAL
  列のデータ型に合わせて値を自動的に変更するかどうか。
- **invalidate**: **boolean** OPTIONAL
  グリッドを無効にして変更を表示するかどうか。

戻り値 **boolean**

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

文字列を行および列に解析し、その内容を特定の範囲に適用します。

非表示の行および列はスキップされます。

パラメーター

- **text**: **string**
  グリッドに解析する、タブと改行で区切られたテキスト。

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

戻り値 **void**

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

指定されたセルの編集を開始します。

FlexGrid の編集は、Excelの編集によく似ています。 ［F2］キーを押すか、セルをダブルクリックすると、 グリッドが**完全編集** モードになります。 このモードでは、ユーザーが［Enter］、［Tab］、または［Esc］キーを押すか、マウスを使用して選択範囲を移動するまで、セルエディタはアクティブのままになります。完全編集モードでは、カーソルキーを押しても、グリッドの編集モードは終了しません。

テキストをセルに直接入力すると、グリッドは**クイック編集**モードになります。 このモードでは、ユーザーがEnter、Tab、Esc、またはいずれかの矢印キーを押すまで、セルエディタはアクティブなままになります。

通常、完全編集モードは既存の値を変更する際に使用します。 クイック編集モードは新しいデータをすばやく入力する際に使用します。

編集中に［F2］キーを押すことで、完全編集モードとクイック編集モードを切り替えることができます。

パラメーター

- **fullEdit**: **boolean** OPTIONAL
  ユーザーがカーソルキーを押したときも編集モードのままにするかどうか。 デフォルトはtrueです。

- **r**: **number** OPTIONAL
  編集する行のインデックス。 デフォルトは現在選択されている行です。

- **c**: **number** OPTIONAL
  編集する列のインデックス。 デフォルトは現在選択されている列です。

- **focus**: **boolean** OPTIONAL
  編集開始時に、エディタにフォーカスを与えるかどうか。 デフォルトはtrueです。

戻り値 **boolean**

toggleDropDownList

```
toggleDropDownList(): void
```

現在選択されているセルに関連付けられたドロップダウンリストボックスを切り替えます。

このメソッドでは、セルが編集モードに入ったとき、またはユーザが特定のキーを押したときに、ドロップダウンリストを自動的に表示することができます。

たとえば、このコードでは、グリッドが編集モードに入るたびに、グリッドにドロップダウンリストが表示されます。

```
// グリッドが編集モードに入ると、ドロップダウンリストを表示する。
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

このコードでは、ユーザーがスペースバーを押した場合、グリッドが編集モードに入った後で、ドロップダウンリストが表示されます。

```
// ユーザーがスペースバーを押したときにドロップダウンリストを表示する。
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

戻り値 **void**

イベント

autoSizedColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定した後に発生します。

引数 **CellRangeEventArgs**

autoSizedRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定した後に発生します。

引数 **CellRangeEventArgs**

autoSizingColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定する前に発生します。

引数 **CellRangeEventArgs**

autoSizingRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定する前に発生します。

引数 **CellRangeEventArgs**

beginningEdit

セルが編集モードに入る前に発生します。

引数 **CellRangeEventArgs**

cellEditEnded

セル編集が確定またはキャンセルされたときに発生します。

引数 **CellRangeEventArgs**

cellEditEnding

セル編集が終了するときに発生します。

このイベントを使用して検証を実行し、無効な編集を防ぐことができます。 たとえば、次のコードは、ユーザーが文字「a」を含まない値を入力できないようにします。このコードは、編集が適用される前に、編集前と編集後の値を取得する方法を示しています。

```
function cellEditEnding (sender, e) {

  // 編集前と編集後の値を取得します
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // newValに「a」が含まれていない場合は、編集をキャンセルします
  e.cancel = newVal.indexOf('a') < 0;
}
```

cancel パラメータをtrueに設定すると、編集された値が無視されて、グリッドでセルの元の値が維持されます。

また、**stayInEditMode** パラメータをtrueに設定すると、グリッドの編集モードが維持され、編集をコミットする前にユーザーが無効な値を修正できます。

引数 **CellEditEndingEventArgs**

copied

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーした後に発生します（**autoClipboard** プロパティを参照）。

引数 **CellRangeEventArgs**

copying

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーするときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラでコピー操作をキャンセルできます。

引数 **CellRangeEventArgs**

deletedRow

ユーザーが［Del］キーを押して行を削除した後に発生します （**allowDelete** プロパティを参照）。

引数 **CellRangeEventArgs**

deletingRow

ユーザーが［Delete］キーを押して選択されている行を削除するときに発生します（**allowDelete** プロパティを参照）。

イベントハンドラで行の削除をキャンセルできます。

引数 **CellRangeEventArgs**

draggedColumn

ユーザーが列のドラッグを完了したときに発生します。

引数 **CellRangeEventArgs**

draggedRow

ユーザーが行のドラッグを完了したときに発生します。

引数 **CellRangeEventArgs**

draggingColumn

ユーザーが列のドラッグを開始するときに発生します。

引数 **CellRangeEventArgs**

draggingColumnOver

ユーザーが列を別の位置にドラッグするときに発生します。

ハンドラは、このイベントをキャンセルして、ユーザーが特定の位置に列をドロップしないようにすることができます。次に例を示します。

```
// ドラッグされている列を記憶します
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// 'sales'列がインデックス0にドラッグされないようにします
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

引数 **CellRangeEventArgs**

draggingRow

ユーザーが行のドラッグを開始するときに発生します。

引数 **CellRangeEventArgs**

draggingRowOver

ユーザーが行を別の位置にドラッグするときに発生します。

引数 **CellRangeEventArgs**

formatItem

セルを表す要素が作成されたときに発生します。

このイベントを使用してセルを表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、グループ行のセルから'wj-wrap'クラスを削除します。

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

groupCollapsedChanged

グループが展開または折りたたまれた後に発生します。

引数 **CellRangeEventArgs**

groupCollapsedChanging

グループが展開または折りたたまれる直前に発生します。

引数 **CellRangeEventArgs**

itemsSourceChanged

グリッドが新しい項目ソースにバインドされた後に発生します。

引数 **EventArgs**

itemsSourceChanging

グリッドが新しい項目ソースにバインドされた後に発生します。

引数 **CancelEventArgs**

loadedRows

グリッド行がデータソースの項目に連結された後に発生します。

引数 **EventArgs**

loadingRows

グリッド行がデータソースの項目に連結される前に発生します。

引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

pasted

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

引数 **CellRangeEventArgs**

pastedCell

ユーザーがクリップボードの内容をセルに貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

引数 **CellRangeEventArgs**

pasting

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた時に発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

引数 **CellRangeEventArgs**

pastingCell

ユーザーがクリップボードの内容をセルに貼り付けるときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

引数 **CellRangeEventArgs**

prepareCellForEdit

エディタセルが作成されたとき、それがアクティブになる前に発生します。

引数 **CellRangeEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

resizedColumn

ユーザーが列のサイズ変更を完了したときに発生します。

引数 **CellRangeEventArgs**

resizedRow

ユーザーが行のサイズ変更を完了したときに発生します。

引数 **CellRangeEventArgs**

resizingColumn

列がサイズ変更されるときに発生します。

引数 **CellRangeEventArgs**

resizingRow

行がサイズ変更されるときに発生します。

引数 **CellRangeEventArgs**

rowAdded

ユーザーが新規行テンプレートを編集して新しい項目を作成したときに発生します（**allowAddNew** プロパティを参照）。

イベントハンドラで新しい項目の内容をカスタマイズしたり、新しい項目の作成をキャンセルしたりできます。

引数 **CellRangeEventArgs**

rowEditEnded

行編集が確定またはキャンセルされたときに発生します。

引数 **CellRangeEventArgs**

rowEditEnding

行編集が終了するとき、変更が確定またはキャンセルされる前に発生します。

このイベントを**rowEditStarted**イベントと組み合わせて使用して、ディープ連結編集を元に戻す操作を実装できます。次に例を示します。

```
// 編集の開始時にディープ連結値を保存します
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.')> -1) { // ディープ連結
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// 編集がキャンセルされたときにディープ連結値を復元します
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // ユーザーによって編集がキャンセルされました
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

引数 **CellRangeEventArgs**

rowEditStarted

行が編集モードに入った後に発生します。

引数 **CellRangeEventArgs**

rowEditStarting

行が編集モードに入る前に発生します。

引数 **CellRangeEventArgs**

scrollPositionChanged

コントロールがスクロールされた後に発生します。

引数 **EventArgs**

selectionChanged

選択が変更された後に発生します。

引数 **CellRangeEventArgs**

selectionChanging

選択が変更される前に発生します。

引数 **CellRangeEventArgs**

sortedColumn

ユーザーが列ヘッダをクリックしてソートを適用した後に発生します。

引数 **CellRangeEventArgs**

sortingColumn

ユーザーが列ヘッダをクリックしてソートを適用する前に発生します。

引数 **CellRangeEventArgs**

updatedLayout

グリッドで内部レイアウトが更新された後に発生します。

引数 **EventArgs**

updatedView

グリッドが現在のビューを構成する要素の作成/更新を完了したときに発生します。

グリッドのビューは以下のような操作に応じて更新されます。

- グリッドまたはそのデータソースの更新
- 行または列の追加、削除、変更
- グリッドのサイズ変更またはスクロール
- 選択の変更

引数 **EventArgs**

updatingLayout

グリッドで内部レイアウトが更新される前に発生します。

引数 **CancelEventArgs**

updatingView

現在のビューを構成する要素の作成/更新をグリッドが開始したときに発生します。

引数 **CancelEventArgs**

FormatItemEventArgs クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
基本クラス **CellRangeEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

formatItem イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- cell
- col
- data
- empty
- panel
- range
- row

コンストラクタ

constructor

```
constructor(p: GridPanel, rng: CellRange, cell: HTMLElement): FormatItemEventArgs
```

FormatItemEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。
- **rng**: **CellRange**
  イベントの影響を受けるセルの範囲。
- **cell**: **HTMLElement**
  書式設定されるグリッドセルを表す要素。

戻り値 **FormatItemEventArgs**

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

cell

書式設定されるグリッドセルを表す要素への参照を取得します。

型 **HTMLElement**

col

このイベントの影響を受けた列を取得します。

継承元 **CellRangeEventArgs**
型 **number**

data

このイベントに関連するデータを取得または設定します。

継承元 **CellRangeEventArgs**
型 **any**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

panel

このイベントの影響を受ける **GridPanel** を取得します。

継承元 **CellRangeEventArgs**
型 **GridPanel**

range

このイベントの影響を受ける **CellRange** を取得します。

継承元 **CellRangeEventArgs**
型 **CellRange**

row

このイベントの影響を受けた行を取得します。

継承元 **CellRangeEventArgs**
型 **number**

GridPanel クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
派生クラス **FlexSheetPanel**

グリッドの論理的な部分（列ヘッダ、行ヘッダ、スクロール可能なデータ部分など）を表します。

コンストラクタ

- constructor

プロパティ

- cellType
- columns
- grid
- height
- hostElement
- rows
- viewRange
- width

メソッド

- getCellBoundingRect
- getCellData
- getCellElement
- getSelectedState
- setCellData

コンストラクタ

constructor

```
constructor(g: FlexGrid, cellType: CellType, rows: RowCollection, cols: ColumnCollection, host: HTMLElement): GridPanel
```

GridPanel クラスの新しいインスタンスを初期化します。

パラメーター

- **g**: **FlexGrid**
  パネルを所有する**FlexGrid** オブジェクト。
- **cellType**: **CellType**
  パネル内のセルのタイプ。
- **rows**: **RowCollection**
  パネルに表示される行。
- **cols**: **ColumnCollection**
  パネルに表示される列。
- **host**: **HTMLElement**
  コントロール内のセルをホストするHTMLElement。

戻り値 **GridPanel**

プロパティ

- cellType

パネルに含まれるセルのタイプを取得します。

型 **CellType**

- columns

パネルの列コレクションを取得します。

型 **ColumnCollection**

- grid

パネルを所有するグリッドを取得します。

型 **FlexGrid**

- height

このパネルに含まれる内容全体の高さを取得します。

型 **number**

- hostElement

パネルのホスト要素を取得します。

型 **HTMLElement**

- rows

パネルの行コレクションを取得します。

型 **RowCollection**

- viewRange

このパネル上の現在表示されているセルの範囲を示す**CellRange** を取得します。

型 **CellRange**

- width

パネルに含まれる内容全体の幅を取得します。

型 **number**

メソッド

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セルの範囲（ビューポート座標単位）を取得します。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

戻り値 **Rect**

getCellData

```
getCellData(r: number, c: any, formatted: boolean): any
```

パネル内のセルに格納された値を取得します。

パラメーター

- **r**: **number**
  セルの行インデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

戻り値 **any**

getCellElement

```
getCellElement(r: number, c: number): HTMLElement
```

この**GridPanel** 内のセルを表す要素を取得します。

セルが現在表示されていない場合、このメソッドはnullを返します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

戻り値 **HTMLElement**

getSelectedState

```
getSelectedState(r: number, c: number, rng: CellRange): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

パラメーター

- **r**: **number**
  検査するセルの行インデックス。
- **c**: **number**
  検査するセルの列インデックス。
- **rng**: **CellRange**
  調べるセルを含む**CellRange** 。

戻り値 **SelectedState**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

パネル内のセルの内容を設定します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **value**: **any**
  セルに格納する値。
- **coerce**: **boolean** OPTIONAL
  列のデータ型に合わせて値を自動的に変更するかどうか。
- **invalidate**: **boolean** OPTIONAL
  グリッドを無効にして変更を表示するかどうか。

戻り値 **boolean**

GroupRow クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
基本クラス **Row**
表示 ☐継承されたメンバー ☐イベント発生元

行グループのヘッダとなる行を表します。

コンストラクタ

- constructor

プロパティ

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- dataItem
- grid
- hasChildren
- height
- index
- isCollapsed
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- level
- multiLine
- pos
- renderHeight
- renderSize
- size
- visible
- visibleIndex
- wordWrap

メソッド

- getCellRange
- getGroupHeader
- onPropertyChanged

コンストラクタ

constructor

```
constructor(): GroupRow
```

GroupRow クラスの新しいインスタンスを初期化します。

戻り値 **GroupRow**

プロパティ

- allowDragging

ユーザーがマウスで行または列を新しい位置に移動できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- allowMerging

行または列にあるセルを結合できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- allowResizing

ユーザーがマウスで行または列をサイズ変更できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- collectionView

この行または列にバインドされた**ICollectionView** を取得します。

継承元 **RowCol**
型 **ICollectionView**

- cssClass

行または列のヘッダ以外のセルをレンダリングするときに使用するCSSクラス名を取得または設定します。

継承元 **RowCol**
型 **string**

- dataItem

項目がバインドされているデータコレクション内の項目を取得または設定します。

継承元 **Row**
型 **any**

- grid

行または列を所有する**FlexGrid** を取得します。

継承元 **RowCol**
型 **FlexGrid**

- hasChildren

グループ行が子行を持つかどうかを示す値を取得します。

型 **boolean**

- height

行または列の高さを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

継承元 **Row**
型 **number**

- index

行または列の親コレクション内でのインデックスを取得します。

継承元 **RowCol**
型 **number**

- isCollapsed

GroupRowが折りたたまれている（子行が表示されていない）か、展開されている（子行が表示されている）かを示す値を取得または設定します。

型 **boolean**

- isContentHtml

この行または列にあるセルがプレーンテキストではなくHTMLコンテンツを含むかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isReadOnly

行または列のセルを編集できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isSelected

行または列が選択されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isVisible

行または列が表示可能で、なおかつ折りたたまれていないかどうかを示す値を取得または設定します。

このプロパティは読み取り専用です。行または列の表示/非表示設定を変更するには、代わりに**visible** プロパティを使用してください。

継承元 **RowCol**
型 **boolean**

- level

GroupRowに関連付けられたグループの階層レベルを取得または設定します。

型 **number**

- multiLine

この行または列にあるセルの内容が改行文字(\n).でラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- pos

行または列の位置を取得します。

継承元 **RowCol**
型 **number**

- renderHeight

行のレンダリング高さを取得します。

列の表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮した幅が返されます。

継承元 **Row**
型 **number**

- renderSize

行または列のレンダリングサイズを取得します。 表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮したサイズが返されます。

継承元 **RowCol**
型 **number**

- size

行または列のサイズを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

継承元 **RowCol**
型 **number**

visible

行または列が表示されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

visibleIndex

非表示にした要素(**isVisible**)を無視し、親コレクション内の行または列のインデックスを取得します。

継承元 **RowCol**
型 **number**

wordWrap

この行または列のセルの内容を使用可能な列幅に収まれるようにラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

メソッド

getCellRange

```
getCellRange(): CellRange
```

この**GroupRow** で表されるグループ内のすべての行およびグリッド内のすべての列を含む **CellRange** オブジェクトを取得します。

戻り値 **CellRange**

getGroupHeader

```
getGroupHeader(): string
```

この**GroupRow** のヘッダテキストを取得します。

戻り値 **string**

onPropertyChanged

```
onPropertyChanged(): void
```

オーナーリストをダーティとしてマークし、オーナーグリッドを更新します。

継承元 **RowCol**
戻り値 **void**

HitTestInfo クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**

指定されたページ座標にある**FlexGrid** コントロールの部分に関する情報を格納します。

コンストラクタ

- constructor

プロパティ

- cellType
- col
- edgeBottom
- edgeLeft
- edgeRight
- edgeTop
- grid
- panel
- point
- range
- row

コンストラクタ

constructor

```
constructor(grid: any, pt: any): HitTestInfo
```

HitTestInfo クラスの新しいインスタンスを初期化します。

パラメーター

- **grid**: **any**
  調査する**FlexGrid** コントロール、**GridPanel** またはセル要素。
- **pt**: **any**
  調査する**Point** オブジェクト（ページ座標単位）。

戻り値 **HitTestInfo**

プロパティ

- cellType

指定された位置にあるセルタイプを取得します。

型 **CellType**

- col

指定された位置にあるセルの列インデックスを取得します。

型 **number**

- edgeBottom

マウスがセルの下端に近いかどうかを示す値を取得します。

型 **boolean**

- edgeLeft

マウスがセルの左端に近いかどうかを示す値を取得します。

型 **boolean**

- edgeRight

マウスがセルの右端に近いかどうかを示す値を取得します。

型 **boolean**

- edgeTop

マウスがセルの上端に近いかどうかを示す値を取得します。

型 **boolean**

- grid

この**HitTestInfo** が参照する**FlexGrid** を取得します。

型 **FlexGrid**

- panel

この**HitTestInfo** が参照する**GridPanel** を取得します。

型 **GridPanel**

- point

この**HitTestInfo** が参照するコントロール座標内のポイントを取得します。

型 **Point**

- range

指定された位置にあるセル範囲を取得します。

型 **CellRange**

- row

指定された位置にあるセルの行インデックスを取得します。

型 **number**

MergeManager クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
派生クラス **DetailMergeManager**

FlexGrid のセル結合動作を定義します。

グリッドのデフォルトの結合動作を実装するために、このクラスのインスタンスが 自動的に作成され、グリッドの **mergeManager** プロパティに割り当てられます。

デフォルトの結合動作をカスタマイズする場合は、 **MergeManager** から派生する クラスを作成し、**getMergedRange** メソッドをオーバーライドします。

コンストラクタ

- constructor

メソッド

- getMergedRange

コンストラクタ

constructor

```
constructor(g: FlexGrid): MergeManager
```

MergeManager クラスの新しいインスタンスを初期化します。

パラメーター

- **g**: **FlexGrid**
  この**MergeManager** を所有する**FlexGrid** オブジェクト。

戻り値 **MergeManager**

メソッド

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

GridPanel 内のセルの結合範囲を示す**CellRange** を取得します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。
- **r**: **number**
  セルが含まれる行のインデックス。
- **c**: **number**
  セルが含まれる列のインデックス。
- **clip**: **boolean** OPTIONAL
  結合範囲をグリッドの現在のビュー範囲にクリップするかどうか。

戻り値 **CellRange**

Row クラス

ファイル	wijmo.grid.js
モジュール	**wijmo.grid**
基本クラス	**RowCol**
派生クラス	**GroupRow**, **DetailRow**, **HeaderRow**
表示	☐継承されたメンバー ☐イベント発生元

グリッドの行を表します。

コンストラクタ

- constructor

プロパティ

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- dataItem
- grid
- height
- index
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- multiLine
- pos
- renderHeight
- renderSize
- size
- visible
- visibleIndex
- wordWrap

メソッド

- onPropertyChanged

コンストラクタ

constructor

```
constructor(dataItem?: any): Row
```

Row クラスの新しいインスタンスを初期化します。

パラメーター

- **dataItem**: **any** OPTIONAL
  この行の連結先であるデータ項目。

戻り値 **Row**

プロパティ

allowDragging

ユーザーがマウスで行または列を新しい位置に移動できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

allowMerging

行または列にあるセルを結合できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

allowResizing

ユーザーがマウスで行または列をサイズ変更できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

collectionView

この行または列にバインドされた**ICollectionView** を取得します。

継承元 **RowCol**
型 **ICollectionView**

cssClass

行または列のヘッダ以外のセルをレンダリングするときに使用するCSSクラス名を取得または設定します。

継承元 **RowCol**
型 **string**

dataItem

項目がバインドされているデータコレクション内の項目を取得または設定します。

型 **any**

- grid

行または列を所有する**FlexGrid** を取得します。

継承元 **RowCol**
型 **FlexGrid**

- height

行または列の高さを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

型 **number**

- index

行または列の親コレクション内でのインデックスを取得します。

継承元 **RowCol**
型 **number**

- isContentHtml

この行または列にあるセルがプレーンテキストではなくHTMLコンテンツを含むかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isReadOnly

行または列のセルを編集できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isSelected

行または列が選択されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isVisible

行または列が表示可能で、なおかつ折りたたまれていないかどうかを示す値を取得または設定します。

このプロパティは読み取り専用です。行または列の表示/非表示設定を変更するには、代わりに**visible** プロパティを使用してください。

継承元 **RowCol**
型 **boolean**

- multiLine

この行または列にあるセルの内容が改行文字(\n).でラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

pos

行または列の位置を取得します。

継承元 **RowCol**
型 **number**

renderHeight

行のレンダリング高さを取得します。

列の表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮した幅が返されます。

型 **number**

renderSize

行または列のレンダリングサイズを取得します。 表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮したサイズが返されます。

継承元 **RowCol**
型 **number**

size

行または列のサイズを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

継承元 **RowCol**
型 **number**

visible

行または列が表示されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

visibleIndex

非表示にした要素(**isVisible**)を無視し、親コレクション内の行または列のインデックスを取得します。

継承元 **RowCol**
型 **number**

wordWrap

この行または列のセルの内容を使用可能な列幅に収まれるようにラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

メソッド

onPropertyChanged

```
onPropertyChanged(): void
```

オーナーリストをダーティとしてマークし、オーナーグリッドを更新します。

継承元 **RowCol**
戻り値 **void**

RowCol クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
派生クラス **Column**, **Row**

Row クラスと**Column** クラスの基本となる抽象クラス。

プロパティ

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- grid
- index
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- multiLine
- pos
- renderSize
- size
- visible
- visibleIndex
- wordWrap

メソッド

- onPropertyChanged

プロパティ

allowDragging

ユーザーがマウスで行または列を新しい位置に移動できるかどうかを示す値を取得または設定します。

型 **boolean**

allowMerging

行または列にあるセルを結合できるかどうかを示す値を取得または設定します。

型 **boolean**

allowResizing

ユーザーがマウスで行または列をサイズ変更できるかどうかを示す値を取得または設定します。

型 **boolean**

- collectionView

この行または列にバインドされた**ICollectionView** を取得します。

型 **ICollectionView**

- cssClass

行または列のヘッダ以外のセルをレンダリングするときに使用するCSSクラス名を取得または設定します。

型 **string**

- grid

行または列を所有する**FlexGrid** を取得します。

型 **FlexGrid**

- index

行または列の親コレクション内でのインデックスを取得します。

型 **number**

- isContentHtml

この行または列にあるセルがプレーンテキストではなくHTMLコンテンツを含むかどうかを示す値を取得または設定します。

型 **boolean**

- isReadOnly

行または列のセルを編集できるかどうかを示す値を取得または設定します。

型 **boolean**

- isSelected

行または列が選択されているかどうかを示す値を取得または設定します。

型 **boolean**

- isVisible

行または列が表示可能で、なおかつ折りたたまれていないかどうかを示す値を取得または設定します。

このプロパティは読み取り専用です。行または列の表示/非表示設定を変更するには、代わりに**visible** プロパティを使用してください。

型 **boolean**

- multiLine

この行または列にあるセルの内容が改行文字(\n).でラップするかどうかを示す値を取得または設定します。

型 **boolean**

- pos

行または列の位置を取得します。

型 **number**

- renderSize

行または列のレンダリングサイズを取得します。 表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮したサイズが返されます。

型 **number**

- size

行または列のサイズを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

型 **number**

- visible

行または列が表示されているかどうかを示す値を取得または設定します。

型 **boolean**

- visibleIndex

非表示にした要素(**`isVisible`**)を無視し、親コレクション内の行または列のインデックスを取得します。

型 **number**

- wordWrap

この行または列のセルの内容を使用可能な列幅に収まれるようにラップするかどうかを示す値を取得または設定します。

型 **boolean**

メソッド

- onPropertyChanged

```
onPropertyChanged(): void
```

オーナーリストをダーティとしてマークし、オーナーグリッドを更新します。

戻り値 **void**

RowColCollection クラス

ファイル	wijmo.grid.js
モジュール	**wijmo.grid**
基本クラス	**ObservableArray**
派生クラス	**ColumnCollection**, **RowCollection**
表示	☐継承されたメンバー ☐イベント発生元

行コレクションと列コレクションの基本となる抽象クラス。

コンストラクタ

- constructor

プロパティ

- defaultSize
- frozen
- isUpdating
- maxSize
- minSize
- visibleLength

メソッド

- beginUpdate
- canMoveElement
- clear
- deferUpdate
- endUpdate
- getItemAt
- getNextCell
- getTotalSize
- implementsInterface
- indexOf
- insert
- isFrozen
- moveElement
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

イベント

- collectionChanged

コンストラクタ

constructor

```
constructor(g: FlexGrid, defaultSize: number): RowColCollection
```

RowColCollection クラスの新しいインスタンスを初期化します。

パラメーター

- **g**: **FlexGrid**
  コレクションを所有する**FlexGrid** 。
- **defaultSize**: **number**
  コレクションの要素のデフォルトサイズ。

戻り値 **RowColCollection**

プロパティ

- defaultSize

コレクションの要素のデフォルトサイズを取得または設定します。

型 **number**

- frozen

コレクションに含まれる静止行または静止列の数を取得または設定します。

静止行および静止列はスクロールされず、グリッドの上または左に（固定セルに隣接して）固定されます。ただし、固定セルとは異なり、静止セルは通常のセルと同じように選択して編集できます。

型 **number**

- isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

継承元 **ObservableArray**
型

- maxSize

コレクションの要素の最大サイズを取得または設定します。

型 **number**

- minSize

コレクションの要素の最小サイズを取得または設定します。

型 **number**

- visibleLength

コレクション内(**isVisible**)に表示される要素の数を取得します。

型 **number**

メソッド

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

戻り値 **void**

canMoveElement

```
canMoveElement(src: number, dst: number): boolean
```

要素をある位置から別の位置に移動できるかどうかをチェックします。

パラメーター

- **src**: **number**
  移動する要素のインデックス。
- **dst**: **number**
  要素の移動先の位置。末尾に移動する場合は-1を指定します。

戻り値 **boolean**

clear

```
clear(): void
```

配列からすべての項目を削除します。

継承元 **ObservableArray**
戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

継承元 **ObservableArray**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **ObservableArray**
戻り値 **void**

getItemAt

```
getItemAt(position: number): number
```

指定した物理位置にある要素のインデックスを取得します。

パラメーター

- **position**: **number**
  コレクションの項目の位置（ピクセル単位）。

戻り値 **number**

getNextCell

```
getNextCell(index: number, move: SelMove, pageSize: number): void
```

選択を変更するために次の表示可能なセルを検索します。

パラメーター

- **index**: **number**
  検索の開始インデックス。

- **move**: **SelMove**
  移動のタイプ（サイズと方向）。

- **pageSize**: **number**
  ページのサイズ（移動がページアップ/ダウンの場合）。

戻り値 **void**

getTotalSize

```
getTotalSize(): number
```

コレクションの要素の合計サイズを取得します。

戻り値 **number**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **ObservableArray**
戻り値 **boolean**

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

配列内で項目を検索します。

パラメーター

- **searchElement**: **any**
  配列内で検索する要素。

- **fromIndex**: **number** OPTIONAL
  検索を開始するインデックス。

継承元 **ObservableArray**
戻り値 **number**

insert

```
insert(index: number, item: any): void
```

配列内の指定した位置に項目を挿入します。

パラメーター

- **index**: **number**
  項目を追加する位置。

- **item**: **any**
  配列に追加する項目。

継承元 **ObservableArray**
戻り値 **void**

isFrozen

```
isFrozen(index: number): boolean
```

行または列が静止行または静止列かどうかをチェックします。

パラメーター

- **index**: **number**
  チェックする行または列のインデックス。

戻り値 **boolean**

moveElement

```
moveElement(src: number, dst: number): void
```

要素をある位置から別の位置に移動します。

パラメーター

- **src**: **number**
  移動する要素のインデックス。

- **dst**: **number**
  要素の移動先を示す位置（末尾に移動する場合は-1）。

戻り値 **void**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

ダーティ状態を追跡し、変更時にグリッドを無効化します。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL

戻り値 **void**

push

```
push(item: any): number
```

配列の最後に項目を追加します。

パラメーター

- **item**: **any**
  配列に追加する項目。

戻り値 **number**

remove

```
remove(item: any): boolean
```

配列から項目を削除します。

パラメーター

- **item**: **any**
  削除する項目。

継承元 **ObservableArray**
戻り値 **boolean**

removeAt

```
removeAt(index: number): void
```

配列内の指定した位置にある項目を削除します。

パラメーター

- **index**: **number**
  削除する項目の位置。

継承元 **ObservableArray**
戻り値 **void**

setAt

```
setAt(index: number, item: any): void
```

配列内の指定した位置にある項目を設定します。

パラメーター

- **index**: **number**
  項目を設定する位置。

- **item**: **any**
  配列に設定する項目。

継承元 **ObservableArray**
戻り値 **void**

slice

```
slice(begin?: number, end?: number): any[]
```

配列の一部のシャローコピーを作成します。

パラメーター

- **begin**: **number** OPTIONAL
  コピーを開始する位置。

- **end**: **number** OPTIONAL
  コピーを終了する位置。

継承元 **ObservableArray**
戻り値 **any[]**

sort

```
sort(compareFn?: Function): this
```

配列の要素を適切な位置にソートします。

パラメーター

- **compareFn**: **Function** OPTIONAL
  ソート順序を定義する関数。 この関数は2つの引数を受け取り、-1（最初の引数の方が2番目の引数より小さい場合）、+1（大きい場合）、0（等しい場合）のいずれかの値を返す必要があります。 これを省略した場合は、各要素の文字列変換に従って辞書順にソートされます。

継承元 `ObservableArray`
戻り値 **this**

splice

```
splice(index: number, count: number, item?: any): any[]
```

配列からの項目の削除、または配列への項目の追加を行います。

パラメーター

- **index**: **number**
  項目を追加または削除する位置。

- **count**: **number**
  配列から削除する項目の数。

- **item**: **any** OPTIONAL
  配列に追加する項目。

戻り値 **any[]**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 `ObservableArray`
引数 `NotifyCollectionChangedEventArgs`

RowCollection クラス

ファイル wijmo.grid.js
モジュール **wijmo.grid**
基本クラス **RowColCollection**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid コントロール内の**Row** オブジェクトのコレクションを表します。

コンストラクタ

- constructor

プロパティ

- ariaLabel
- defaultSize
- frozen
- isUpdating
- maxGroupLevel
- maxSize
- minSize
- visibleLength

メソッド

- beginUpdate
- canMoveElement
- clear
- deferUpdate
- endUpdate
- getItemAt
- getNextCell
- getTotalSize
- implementsInterface
- indexOf
- insert
- isFrozen
- moveElement
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

イベント

- collectionChanged

コンストラクタ

constructor

```
constructor(g: FlexGrid, defaultSize: number): RowColCollection
```

RowColCollection クラスの新しいインスタンスを初期化します。

パラメーター

- **g**: **FlexGrid**
  コレクションを所有する**FlexGrid** 。

- **defaultSize**: **number**
  コレクションの要素のデフォルトサイズ。

継承元 **RowColCollection**
戻り値 **RowColCollection**

プロパティ

ariaLabel

このコレクション内のすべての行に対してARIAラベルとして使用される文字列を取得または設定します。

たとえば、次のコードは、ARIAラベルをヘッダ行とデータ行に追加します。

```
grid.rows.ariaLabel = 'data row';
grid.columnHeaders.rows.ariaLabel = 'header row';
```

型 **string**

defaultSize

コレクションの要素のデフォルトサイズを取得または設定します。

継承元 **RowColCollection**
型 **number**

frozen

コレクションに含まれる静止行または静止列の数を取得または設定します。

静止行および静止列はスクロールされず、グリッドの上または左に（固定セルに隣接して）固定されます。ただし、固定セルとは異なり、静止セルは通常のセルと同じように選択して編集できます。

継承元 **RowColCollection**
型 **number**

isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

継承元 **ObservableArray**
型

maxGroupLevel

グリッドの最大グループレベルを取得します。

型 **number**

maxSize

コレクションの要素の最大サイズを取得または設定します。

継承元 **RowColCollection**
型 **number**

minSize

コレクションの要素の最小サイズを取得または設定します。

継承元 **RowColCollection**
型 **number**

visibleLength

コレクション内(**isVisible**)に表示される要素の数を取得します。

継承元 **RowColCollection**
型 **number**

メソッド

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **RowColCollection**
戻り値 **void**

canMoveElement

```
canMoveElement(src: number, dst: number): boolean
```

要素をある位置から別の位置に移動できるかどうかをチェックします。

パラメーター

- **src**: **number**
  移動する要素のインデックス。
- **dst**: **number**
  要素の移動先の位置。末尾に移動する場合は-1を指定します。

継承元 **RowColCollection**
戻り値 **boolean**

clear

```
clear(): void
```

配列からすべての項目を削除します。

継承元 **ObservableArray**
戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

継承元 **ObservableArray**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **ObservableArray**
戻り値 **void**

getItemAt

```
getItemAt(position: number): number
```

指定した物理位置にある要素のインデックスを取得します。

パラメーター

- **position**: **number**
  コレクションの項目の位置（ピクセル単位）。

継承元 **RowColCollection**
戻り値 **number**

getNextCell

```
getNextCell(index: number, move: SelMove, pageSize: number): void
```

選択を変更するために次の表示可能なセルを検索します。

パラメーター

- **index**: **number**
  検索の開始インデックス。
- **move**: **SelMove**
  移動のタイプ（サイズと方向）。
- **pageSize**: **number**
  ページのサイズ（移動がページアップ/ダウンの場合）。

継承元 **RowColCollection**
戻り値 **void**

getTotalSize

```
getTotalSize(): number
```

コレクションの要素の合計サイズを取得します。

継承元 **RowColCollection**
戻り値 **number**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **ObservableArray**
戻り値 **boolean**

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

配列内で項目を検索します。

パラメーター

- **searchElement**: **any**
  配列内で検索する要素。
- **fromIndex**: **number** OPTIONAL
  検索を開始するインデックス。

継承元 **ObservableArray**
戻り値 **number**

insert

```
insert(index: number, item: any): void
```

配列内の指定した位置に項目を挿入します。

パラメーター

- **index**: **number**
  項目を追加する位置。

- **item**: **any**
  配列に追加する項目。

継承元 **ObservableArray**
戻り値 **void**

isFrozen

```
isFrozen(index: number): boolean
```

行または列が静止行または静止列かどうかをチェックします。

パラメーター

- **index**: **number**
  チェックする行または列のインデックス。

継承元 **RowColCollection**
戻り値 **boolean**

moveElement

```
moveElement(src: number, dst: number): void
```

要素をある位置から別の位置に移動します。

パラメーター

- **src**: **number**
  移動する要素のインデックス。

- **dst**: **number**
  要素の移動先を示す位置（末尾に移動する場合は-1）。

継承元 **RowColCollection**
戻り値 **void**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

ダーティ状態を追跡し、変更時にグリッドを無効化します。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL

継承元 **RowColCollection**
戻り値 **void**

push

```
push(item: any): number
```

配列の最後に項目を追加します。

パラメーター

- **item**: **any**
  配列に追加する項目。

継承元 **RowColCollection**
戻り値 **number**

remove

```
remove(item: any): boolean
```

配列から項目を削除します。

パラメーター

- **item**: **any**
  削除する項目。

継承元 **ObservableArray**
戻り値 **boolean**

removeAt

```
removeAt(index: number): void
```

配列内の指定した位置にある項目を削除します。

パラメーター

- **index**: **number**
  削除する項目の位置。

継承元 **ObservableArray**
戻り値 **void**

setAt

```
setAt(index: number, item: any): void
```

配列内の指定した位置にある項目を設定します。

パラメーター

- **index**: **number**
  項目を設定する位置。

- **item**: **any**
  配列に設定する項目。

継承元 **ObservableArray**
戻り値 **void**

slice

```
slice(begin?: number, end?: number): any[]
```

配列の一部のシャローコピーを作成します。

パラメーター

- **begin**: **number** OPTIONAL
  コピーを開始する位置。

- **end**: **number** OPTIONAL
  コピーを終了する位置。

継承元 **ObservableArray**
戻り値 **any[]**

sort

```
sort(compareFn?: Function): this
```

配列の要素を適切な位置にソートします。

パラメーター

- **compareFn**: **Function** OPTIONAL
  ソート順序を定義する関数。 この関数は2つの引数を受け取り、-1（最初の引数の方が2番目の引数より小さい場合）、+1（大きい場合）、0（等しい場合）のいずれかの値を返す必要があります。 これを省略した場合は、各要素の文字列変換に従って辞書順にソートされます。

継承元 **ObservableArray**
戻り値 **this**

splice

```
splice(index: number, count: number, item?: any): any[]
```

配列からの項目の削除、または配列への項目の追加を行います。

パラメーター

- **index**: **number**
  項目を追加または削除する位置。

- **count**: **number**
  配列から削除する項目の数。

- **item**: **any** OPTIONAL
  配列に追加する項目。

継承元 **RowColCollection**
戻り値 **any[]**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 **ObservableArray**
引数 **NotifyCollectionChangedEventArgs**

AllowDragging 列挙体

ファイル `wijmo.grid.js`
モジュール **wijmo.grid**

行/列のドラッグ動作を定義する定数を指定します。

メンバー

名前	値	説明
None	0	ユーザーは行または列をドラッグできません。
Columns	1	ユーザーは列をドラッグできます。
Rows	2	ユーザーは行をドラッグできます。
Both	Rows | Columns	ユーザーは行または列をドラッグできます。

AllowMerging 列挙体

ファイル wijmo.grid.js
モジュール **wijmo.grid**

グリッドのどの領域がセル結合をサポートするかを定義する定数を指定します。

メンバー

名前	値	説明
None	0	結合しません。
Cells	1	スクロール可能なセルを結合します。
ColumnHeaders	2	列ヘッダーを結合します。
RowHeaders	4	行ヘッダーを結合します。
AllHeaders	ColumnHeaders | RowHeaders	列ヘッダーと行ヘッダーを結合します。
All	Cells | AllHeaders	すべての領域を結合します。

AllowResizing 列挙体

ファイル `wijmo.grid.js`
モジュール **wijmo.grid**

行/列のサイズ変更動作を定義する定数を指定します。

メンバー

名前	値	説明
None	0	ユーザーは行または列をサイズ変更できません。
Columns	1	ユーザーは、列ヘッダーの端をドラッグすることで、列をサイズ変更できます。
Rows	2	ユーザーは、行ヘッダーの端をドラッグすることで、行をサイズ変更できます。
Both	Rows I Columns	ユーザーは、ヘッダーの端をドラッグすることで、行および列をサイズ変更できます。
ColumnsAllCells	Columns I _AR_ALLCELLS	ユーザーは、任意のセルの端をドラッグすることで、列をサイズ変更できます。
RowsAllCells	Rows I _AR_ALLCELLS	ユーザーは、任意のセルの端をドラッグすることで、行をサイズ変更できます。
BothAllCells	Both I _AR_ALLCELLS	ユーザーは、任意のセルの端をドラッグすることで、行および列をサイズ変更できます。

AutoSizeMode 列挙体

ファイル `wijmo.grid.js`
モジュール **wijmo.grid**

行/列の自動サイズ変更動作を定義する定数を指定します。

メンバー

名前	値	説明
None	0	自動サイズ変更は無効です。
Headers	1	自動サイズ変更はヘッダーセルに機能します。
Cells	2	自動サイズ変更はデータセルに機能します。
Both	Headers | Cells	自動サイズ変更はヘッダーセルとデータセルに機能します。

CellType 列挙体

ファイル `wijmo.grid.js`
モジュール **wijmo.grid**

GridPanel 内のセルのタイプを定義する定数を指定します。

メンバー

名前	値	説明
None	0	不明または無効なセルタイプ。
Cell	1	通常のデータセル
ColumnHeader	2	列ヘッダセル。
RowHeader	3	行ヘッダセル。
TopLeft	4	左上セル。
ColumnFooter	5	列フッターセル。
BottomLeft	6	左下のセル（行ヘッダーと列フッターが交差する位置にあるセル）。

HeadersVisibility 列挙体

ファイル wijmo.grid.js
モジュール **wijmo.grid**

行ヘッダーと列ヘッダーの表示を定義する定数を指定します。

メンバー

名前	値	説明
None	0	ヘッダセルは表示されません。
Column	1	列ヘッダセルのみが表示されます。
Row	2	行ヘッダセルのみが表示されます。
All	3	列ヘッダセルと行ヘッダセルの両方が表示されます。

KeyAction 列挙体

ファイル `wijmo.grid.js`
モジュール **wijmo.grid**

ENTERやTABキーなどの特殊キーが押されたときに実行されるアクションを定義する定数を指定します。

メンバー

名前	値	説明
None	0	特別なアクションの必要はありません（ブラウザがキーを処理します）。
MoveDown	1	選択範囲を次の行に移動します。
MoveAcross	2	選択範囲を次の列に移動します。
Cycle	3	次の列に移動し、次の行に折り返します。
CycleOut	4	次の列に移動し、次の行に折り返してからコントロールの外に移動します。

RowColFlags 列挙体

ファイル wijmo.grid.js
モジュール **wijmo.grid**

グリッド行または列の状態を表すフラグを指定します。

メンバー

名前	値	説明
Visible	1	行または列は表示可能です。
AllowResizing	2	行または列はサイズ変更できます。
AllowDragging	4	行または列はマウスで新しい位置にドラッグできます。
AllowMerging	8	行または列は結合セルを含むことができます。
AllowSorting	16	列ヘッダをマウスでクリックして列をソートできます。
AutoGenerated	32	列は自動的に生成されました。
Collapsed	64	グループ行が折りたたまれています。
ParentCollapsed	128	グループ行が折りたたまれています。
Selected	256	行または列は選択されています。
ReadOnly	512	行または列は読み取り専用です（編集できません）。
HtmlContent	1024	この行または列のセルにHTMLテキストが含まれます。
WordWrap	2048	この行または列のセルに折り返されたテキストを含めることができます。
MultiLine	4096	この行または列のセルに折り返されたテキストを含めることができます。
HasTemplate	8192	この列のセルにはテンプレートが適用されています。
RowDefault	Visible | AllowResizing	新しい行のデフォルト設定。
ColumnDefault	Visible | AllowDragging | AllowResizing | AllowSorting	新しい列のデフォルト設定。

SelectedState 列挙体

ファイル wijmo.grid.js
モジュール **wijmo.grid**

セルの選択状態を表す定数を指定します。

メンバー

名前	値	説明
None	0	セルは選択されていません。
Selected	1	セルは選択されていますが、アクティブセルではありません。
Cursor	2	セルは選択されており、アクティブセルです。
Active	3	セルはアクティブな状態ですが、選択状態ではありません。

SelectionMode 列挙体

ファイル wijmo.grid.js
モジュール **wijmo.grid**

選択動作を定義する定数を指定します。

メンバー

名前	値	説明
None	0	ユーザーは、マウスやキーボードでセルを選択できません。
Cell	1	ユーザーは、一度に1つのセルだけを選択できます。
CellRange	2	ユーザーは、隣接するセルのブロックを選択できません。
Row	3	ユーザーは、一度に1つの行を選択できます。
RowRange	4	ユーザーは、隣接する行を選択できます。
ListBox	5	ユーザーは、隣接するセルのブロックを選択できません。

SelMove 列挙体

ファイル wijmo.grid.js
モジュール **wijmo.grid**

選択範囲の移動の種類を表す定数を指定します。

メンバー

名前	値	説明
None	0	選択範囲を変更しません。
Next	1	表示されている次のセルを選択します。
Prev	2	表示されている前のセルを選択します。
NextPage	3	次のページに表示されている最初のセルを選択します。
PrevPage	4	前のページに表示されている最初のセルを選択します。
Home	5	表示されている最初のセルを選択します。
End	6	表示されている最後のセルを選択します。
NextCell	7	必要な場合は行をスキップし、表示されている次のセルを選択します。
PrevCell	8	必要な場合は行をスキップし、表示されている前のセルを選択します。

wijmo.grid.filter モジュール

ファイル `wijmo.grid.filter.js`
モジュール **wijmo.grid.filter**

FlexGrid コントロールでExcelスタイルのフィルタを提供する拡張です。

クラス

- ColumnFilter
- ColumnFilterEditor
- ConditionFilter
- ConditionFilterEditor
- FilterCondition
- FlexGridFilter
- ValueFilter
- ValueFilterEditor

インターフェイス

- IColumnFilter

列挙体

- FilterType
- Operator

ColumnFilter クラス

ファイル	`wijmo.grid.filter.js`
モジュール	**wijmo.grid.filter**
派生クラス	**FlexSheetColumnFilter**
インターフェイス	**IColumnFilter**

FlexGrid コントロールの列のフィルタを定義します。

ColumnFilter には**ConditionFilter** と**ValueFilter** が含まれます。一度にアクティブにできるのはどちらか一方だけです。

このクラスは**FlexGridFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- column
- conditionFilter
- dataMap
- filterType
- isActive
- valueFilter

メソッド

- apply
- clear
- implementsInterface

コンストラクタ

constructor

```
constructor(owner: FlexGridFilter, column: Column): ColumnFilter
```

ColumnFilter クラスの新しいインスタンスを初期化します。

パラメーター

- **owner**: **FlexGridFilter**
  この列フィルタを所有する**FlexGridFilter** 。
- **column**: **Column**
  フィルタ対象の**Column** 。

戻り値 **ColumnFilter**

プロパティ

column

フィルタリングする**Column** を取得します。

型 **Column**

conditionFilter

このColumnFilter のConditionFilter を取得します。

型 **ConditionFilter**

dataMap

未加工の値をこのフィルタを編集する際に表示される表示値に変換するために使用されるDataMap を取得または設定します。

次の例では、DataMap をBoolean型の列フィルタに割り当て、 フィルタエディタに、'true'と'false'ではなく'Yes'と'No'が表示されるようにしています。

```
var filter = new wijmo.grid.filter.FlexGridFilter(grid),
    map = new wijmo.grid.DataMap([
            { value: true, caption: 'Yes' },
            { value: false, caption: 'No' },
        ], 'value', 'caption');
for (var c = 0; c < grid.columns.length; c++) {
    if (grid.columns[c].dataType == wijmo.DataType.Boolean) {
        filter.getColumnFilter(c).dataMap = map;
    }
}
```

型 **DataMap**

filterType

このフィルタから提供されるフィルタ処理のタイプを取得または設定します。

このプロパティをnullに設定すると、フィルタは、オーナーフィルタの **defaultFilterType** プロパティで定義された値を使用します。

型 **FilterType**

isActive

このフィルタがアクティブかどうかを示す値を取得します。

型 **boolean**

valueFilter

このColumnFilter のValueFilter を取得します。

型 **ValueFilter**

メソッド

apply

```
apply(value): boolean
```

値がフィルタに一致するかどうかを示す値を取得します。

パラメーター

- **value**:
  テストする値。

戻り値 **boolean**

clear

```
clear(): void
```

フィルタをクリアします。

戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

戻り値 **boolean**

ColumnFilterEditor クラス

ファイル	wijmo.grid.filter.js
モジュール	**wijmo.grid.filter**
基本クラス	**Control**
派生クラス	**FlexSheetColumnFilterEditor**
表示	☐ 継承されたメンバー ☐ イベント発生元

列フィルタの設定の確認や変更に使用されるエディタ。

このクラスは**FlexGridFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onButtonClicked
- onFilterChanged
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

イベント

- buttonClicked
- filterChanged
- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, filter: ColumnFilter, sortButtons?: boolean): ColumnFilterEditor
```

ColumnFilterEditor クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **filter**: **ColumnFilter**
  編集する**ColumnFilter** 。
- **sortButtons**: **boolean** OPTIONAL
  エディタにソートボタンを表示するかどうか。

戻り値 **ColumnFilterEditor**

プロパティ

STATIC controlTemplate

ColumnFilterEditor コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

filter

編集する**ColumnFilter** への参照を取得します。

型 **ColumnFilter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**`Control`** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **`removeEventListener`** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **`Improving scrolling performance with passive listeners`**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 `Control`
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onButtonClicked

```
onButtonClicked(e?: EventArgs): void
```

buttonClicked イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onFilterChanged

```
onFilterChanged(e?: EventArgs): void
```

filterChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

updateEditor

```
updateEditor(): void
```

現在のフィルタ設定でエディタを更新します。

戻り値 **void**

updateFilter

```
updateFilter(): void
```

現在のエディタ設定でフィルタを更新します。

戻り値 **void**

イベント

buttonClicked

いずれかのエディタボタンがクリックされたときに発生します。

引数 **EventArgs**

filterChanged

フィルタが変更された後に発生します。

引数 **EventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

ConditionFilter クラス

ファイル	`wijmo.grid.filter.js`
モジュール	**wijmo.grid.filter**
派生クラス	**FlexSheetConditionFilter**
インターフェイス	**IColumnFilter**

FlexGrid コントロールの列の条件フィルタを定義します。

条件フィルタには、'and'または'or'演算子を使用して結合できる2つの条件が含まれます。

このクラスは**FlexGridFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- and
- column
- condition1
- condition2
- dataMap
- isActive

メソッド

- apply
- clear
- implementsInterface

コンストラクタ

constructor

```
constructor(column: Column): ConditionFilter
```

ConditionFilter クラスの新しいインスタンスを初期化します。

パラメーター

- **column**: **Column**
  フィルタリングする列。

戻り値 **ConditionFilter**

プロパティ

- and

2つの条件をAND演算子とOR演算子のどちらを使用して結合するかを示す値を取得します。 The default value for this property is **true**.

型 **boolean**

- column

フィルタリングする**Column** を取得します。

型 **Column**

- condition1

このフィルタの最初の条件を取得します。

型 **FilterCondition**

- condition2

このフィルタの2番目の条件を取得します。

型 **FilterCondition**

- dataMap

未加工の値をこのフィルタを編集する際に表示される表示値に変換するために使用される**DataMap** を取得または設定します。

型 **DataMap**

- isActive

このフィルタがアクティブかどうかを示す値を取得します。

2つの条件のうち少なくとも1つの条件の演算子と値が有効な組み合わせに設定されている場合、そのフィルタはアクティブです。

型 **boolean**

メソッド

- apply

```
apply(value): boolean
```

値がこのフィルタに一致するかどうかを示す値を返します。

パラメーター

- **value**:
  テストする値。

戻り値 **boolean**

- clear

```
clear(): void
```

フィルタをクリアします。

戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

戻り値 **boolean**

ConditionFilterEditor クラス

ファイル wijmo.grid.filter.js
モジュール **wijmo.grid.filter**
基本クラス **Control**
表示 ☐継承されたメンバー ☐イベント発生元

ConditionFilter オブジェクトの設定の確認や変更に使用されるエディタ。

このクラスは**FlexGridFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- clearEditor
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, filter: ConditionFilter): ConditionFilterEditor
```

ConditionFilterEditor クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のセレクター（例: '#theCtrl'）。

- **filter**: **ConditionFilter**
  編集する**ConditionFilter** 。

戻り値 **ConditionFilterEditor**

プロパティ

- STATIC controlTemplate

ConditionFilterEditor コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

- filter

編集する**ConditionFilter** への参照を取得します。

型 **ConditionFilter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**`Control`** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**`dispose`** と **`removeEventListener`** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **`Improving scrolling performance with passive listeners`**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 `Control`
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

clearEditor

```
clearEditor(): void
```

フィルタに変更を適用せずにエディタをクリアします。

戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能性があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

updateEditor

```
updateEditor(): void
```

現在のフィルタ設定でエディタを更新します。

戻り値 **void**

updateFilter

```
updateFilter(): void
```

現在のエディタ値を反映するようにフィルタを更新します。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

FilterCondition クラス

ファイル wijmo.grid.filter.js
モジュール **wijmo.grid.filter**

フィルタ条件を定義します。

このクラスは**FlexGridFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- isActive
- operator
- value

メソッド

- apply
- clear

コンストラクタ

constructor

```
constructor(filter?: ConditionFilter): FilterCondition
```

FilterCondition クラスの新しいインスタンスを初期化します。

パラメーター

- **filter**: **ConditionFilter** OPTIONAL
  この**FilterCondition** を所有する**ConditionFilter** 。

戻り値 **FilterCondition**

プロパティ

- isActive

この条件がアクティブかどうかを示す値を取得します。

型 **boolean**

- operator

この@FilterConditionによって使用される演算子を取得または設定します。

型 **Operator**

- value

この@FilterConditionによって使用される値を取得または設定します。

型 **any**

メソッド

apply

```
apply(value): boolean
```

指定された値がこの**FilterCondition** に適合するかどうかを判定する値を返します。

パラメーター

- **value**:
  テストする値。

戻り値　　**boolean**

clear

```
clear(): void
```

条件をクリアします。

戻り値　　**void**

FlexGridFilter クラス

ファイル wijmo.grid.filter.js
モジュール **wijmo.grid.filter**
派生クラス **FlexSheetFilter**, **WjFlexGridFilter**

FlexGrid コントロールのExcelスタイルのフィルタを実装します。

FlexGrid コントロールでフィルタリングを有効にするには、 **FlexGridFilter** のインスタンスを作成し、コンストラクターのパラメーターとしてグリッドを渡します。例:

```
// FlexGridを作成します。
var flex = new wijmo.grid.FlexGrid('#gridElement');

// FlexGridでフィルタリングを有効にします。
var filter = new wijmo.grid.filter.FlexGridFilter(flex);
```

そうすると、グリッドの列ヘッダにフィルタアイコンが追加されます。 このアイコンをクリックするとエディタが表示され、そこでその列のフィルタ条件を編集できます。

FlexGridFilter クラスは**wijmo.grid**および**wijmo.input**モジュールに依存します。

次の例では、 **FlexGridFilter** を使用して**FlexGrid** コントロールにフィルタ処理を追加する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/1ttsyag4)

The **FlexGridFilter** class relies on the filtering functionality provided by the **CollectionView** class, and therefore can be used only with bound grids.

コンストラクタ

- constructor

プロパティ

- activeEditor
- defaultFilterType
- filterColumns
- filterDefinition
- grid
- showFilterIcons
- showSortButtons

メソッド

- apply
- clear
- closeEditor
- editColumnFilter
- getColumnFilter
- onFilterApplied
- onFilterChanged
- onFilterChanging

イベント

- filterApplied
- filterChanged
- filterChanging

コンストラクタ

constructor

```
constructor(grid: FlexGrid, options?: any): FlexGridFilter
```

FlexGridFilter クラスの新しいインスタンスを初期化します。

パラメーター

- **grid**: **FlexGrid**
  フィルタ対象の**FlexGrid** 。
- **options**: **any** OPTIONAL
  FlexGridFilter の初期化オプション。

戻り値 **FlexGridFilter**

プロパティ

activeEditor

アクティブな**ColumnFilterEditor** を取得します。

このプロパティを使用すると、**filterChanging** イベントを処理するときにフィルターエディターをカスタマイズできます。 フィルターが編集されていない場合はnullを返します。

型 **ColumnFilterEditor**

defaultFilterType

使用するデフォルトフィルタタイプを取得または設定します。

この値は特定の列のフィルタでオーバーライドできます。 たとえば、以下のサンプルコードは、"ByValue"列を除くすべての列に対して条件に基づくフィルタを作成します。

```
var f = new wijmo.grid.filter.FlexGridFilter(flex);
f.defaultFilterType = wijmo.grid.filter.FilterType.Condition;
var col = flex.columns.getColumn('ByValue'),
    cf = f.getColumnFilter(col);
cf.filterType = wijmo.grid.filter.FilterType.Value;
```

The default value for this property is **FilterType.Both**.

型 **FilterType**

filterColumns

フィルタを持つ列の名前またはバインディングを含む配列を取得または設定します。

このプロパティをnullまたは空の配列に設定すると、すべての列にフィルタが追加されます。

型 **string**[]

filterDefinition

現在のフィルタ定義をJSON文字列として取得または設定します。

型 **string**

grid

このフィルタを所有する**FlexGrid** への参照を取得します。

型 **FlexGrid**

showFilterIcons

FlexGridFilter がグリッドの列ヘッダにフィルタ編集ボタンを追加するかどうかを示す値を取得または設定します。

このプロパティをfalseに設定した場合は、ユーザーがフィルタを編集、クリア、および適用する手段を開発者が提供する必要があります。

The default value for this property is **true**.

型 **boolean**

showSortButtons

フィルタエディタにソートボタンが表示されるかどうかを示す値を取得または設定します。

デフォルトでは、エディタにはExcelと同じようにソートボタンが表示されます。しかし、ユーザーはヘッダをクリックすることによって列をソートできるので、 フィルタエディタにソートボタンがあるのは望ましくない場合があります。

The default value for this property is **true**.

型 **boolean**

メソッド

apply

```
apply(): void
```

現在の列フィルタをグリッドに適用します。

戻り値 **void**

clear

```
clear(): void
```

すべての列フィルタをクリアします。

戻り値 **void**

closeEditor

```
closeEditor(): void
```

フィルタエディタを閉じます。

戻り値 **void**

editColumnFilter

```
editColumnFilter(col: any, ht?: HitTestInfo, refElem?: HTMLElement): void
```

指定したグリッド列のフィルタエディタを表示します。

パラメーター

- **col**: **any**
  編集するフィルタを含む**Column** 。
- **ht**: **HitTestInfo** OPTIONAL
  フィルタ表示をトリガしたセルの範囲を含む**HitTestInfo** オブジェクト。
- **refElem**: **HTMLElement** OPTIONAL
  An HTMLElement to use as a reference for positioning the editor.

戻り値 **void**

getColumnFilter

```
getColumnFilter(col: any, create?: boolean): ColumnFilter
```

指定した列のフィルタを取得します。

パラメーター

- **col**: **any**
  フィルタの適用先の**Column** （または列名またはインデックス）。
- **create**: **boolean** OPTIONAL
  存在しない場合にフィルタを作成するかどうか。

戻り値 **ColumnFilter**

onFilterApplied

```
onFilterApplied(e?: EventArgs): void
```

filterApplied イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onFilterChanged

```
onFilterChanged(e: CellRangeEventArgs): void
```

filterChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**

戻り値 **void**

onFilterChanging

```
onFilterChanging(e: CellRangeEventArgs): boolean
```

filterChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

戻り値 **boolean**

イベント

filterApplied

フィルタが適用された後に発生します。

引数 **EventArgs**

filterChanged

ユーザーが列フィルタを編集した後で発生します。

イベントパラメータを使用して、フィルタを所有する列を判定し、 変更が適用されたかキャンセルされたかを判定します。

引数 **CellRangeEventArgs**

filterChanging

ユーザーが列フィルタを編集しようとしたときに発生します。

フィルタのデフォルトの設定をオーバーライドする場合は、 このイベントを使用して列フィルタをカスタマイズします。

たとえば、以下のコードは、フィルタ条件がnullの場合に、使用される演算子を 'contains'に設定します。

```
filter.filterChanging.addHandler(function (s, e) {
  var cf = filter.getColumnFilter(e.col);
  if (!cf.valueFilter.isActive && cf.conditionFilter.condition1.operator == null) {
    cf.filterType = wijmo.grid.filter.FilterType.Condition;
    cf.conditionFilter.condition1.operator = wijmo.grid.filter.Operator.CT;
  }
});
```

引数 **CellRangeEventArgs**

ValueFilter クラス

ファイル	`wijmo.grid.filter.js`
モジュール	**wijmo.grid.filter**
派生クラス	**FlexSheetValueFilter**
インターフェイス	**IColumnFilter**

FlexGrid コントロールの列の値フィルタを定義します。

値フィルタには、グリッドに表示する値の明示的なリストが含まれます。

コンストラクタ

- constructor

プロパティ

- column
- dataMap
- filterText
- isActive
- maxValues
- showValues
- sortValues
- uniqueValues

メソッド

- apply
- clear
- implementsInterface

コンストラクタ

constructor

```
constructor(column: Column): ValueFilter
```

ValueFilter クラスの新しいインスタンスを初期化します。

パラメーター

- **column**: **Column**
  フィルタリングする列。

戻り値 **ValueFilter**

プロパティ

- column

フィルタリングする**Column** を取得します。

型 **Column**

dataMap

未加工の値をこのフィルタを編集する際に表示される表示値に変換するために使用される**DataMap** を取得または設定します。

型 **DataMap**

filterText

表示値のリストのフィルタリングに使用される文字列を取得または設定します。

型 **string**

isActive

このフィルタがアクティブかどうかを示す値を取得します。

少なくとも1つの値が選択されている場合、そのフィルタはアクティブです。

型 **boolean**

maxValues

表示値のリスト内にある要素の最大数を取得または設定します。

非常に多くの項目をリストに追加すると、検索が困難になり、パフォーマンスが損なわれます。このプロパティは任意の時点で表示される項目数を制限しますが、 ユーザーは検索ボックスを使用して目的の項目をフィルタ処理することができます。 このプロパティは、デフォルトでは250に設定されます。

次のコードは、この値を1,000,000に変更し、事実上、フィールドのすべての一意の値を一覧します。

```
// 'id'列のmaxItemsプロパティを変更します。
var f = new wijmo.grid.filter.FlexGridFilter(s);
f.getColumnFilter('id').valueFilter.maxValues = 1000000;
```

型 **number**

showValues

値リストに表示されるすべての書式設定された値を含むオブジェクトを取得または設定します。

型 **any**

sortValues

エディタに表示するときに値をソートするかどうかを決定する値を取得または設定します。

uniqueValues を使用して値のカスタムリストを提供した上、 それらの値の順序を維持したい場合、このプロパティは特に便利です。

型 **boolean**

- uniqueValues

リストに表示する一意の値を含む配列を取得または設定します。

このプロパティがnullに設定されている場合、リストには、グリッドデータに基づいて値が挿入されます。

データからリストを作成するより、一意の値のリストを明示的に割り当てる方が 効率的です。また、データがサーバー上でフィルタ処理される場合、値フィルタが 適切に動作するには、そうする必要があります（この場合、一部の値が クライアント上に存在しない可能性があり、リストが不完全になるため）。

デフォルトでは、フィルタエディタは、一意の値をユーザーに表示する際に値をソートします。この動作を抑止して、指定した順序で値を表示する場合は、 **sortValues** プロパティをfalseに設定します。

たとえば、次のコードは、**ValueFilter** で'country'フィールドに連結された列に 使用される国名リストを提供します。

```
// FlexGridのフィルタを作成します
var filter = new wijmo.grid.filter.FlexGridFilter(grid);

// 一意の値のリストを国フィルタに割り当てます
var cf = filter.getColumnFilter('country');
cf.valueFilter.uniqueValues = countries;
```

型 **any**[]

メソッド

- apply

```
apply(value): boolean
```

値がフィルタに一致するかどうかを示す値を取得します。

パラメーター

- **value**:
  テストする値。

戻り値 **boolean**

- clear

```
clear(): void
```

フィルタをクリアします。

戻り値 **void**

- implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

戻り値 **boolean**

ValueFilterEditor クラス

ファイル	`wijmo.grid.filter.js`
モジュール	**wijmo.grid.filter**
基本クラス	**Control**
派生クラス	**FlexSheetValueFilterEditor**
表示	☐継承されたメンバー ☐イベント発生元

ValueFilter オブジェクトの設定の確認や変更に使用されるエディタ。

このクラスは**FlexGridFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- clearEditor
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, filter: ValueFilter): ValueFilterEditor
```

ValueFilterEditor クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のセレクター（例: '#theCtrl'）。

- **filter**: **ValueFilter**
  編集する**ValueFilter** 。

戻り値 **ValueFilterEditor**

プロパティ

- STATIC controlTemplate

ColumnFilterEditor コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

- filter

編集する**ValueFilter** への参照を取得します。

型 **ValueFilter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

clearEditor

```
clearEditor(): void
```

フィルタに変更を適用せずにエディタをクリアします。

戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能性があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

updateEditor

```
updateEditor(): void
```

現在のフィルタ設定でエディタを更新します。

戻り値 **void**

updateFilter

```
updateFilter(): void
```

現在のエディタ値を反映するようにフィルタを更新します。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

IColumnFilter インターフェイス

ファイル wijmo.grid.filter.js
モジュール **wijmo.grid.filter**

FlexGrid コントロールの列のフィルタを定義します。

このクラスは**FlexGridFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

FilterType 列挙体

ファイル `wijmo.grid.filter.js`
モジュール **wijmo.grid.filter**

列フィルタのタイプを指定します。

メンバー

名前	値	説明
None	0	フィルタなし。
Condition	1	2つの条件に基づくフィルタ。
Value	2	値のセットに基づくフィルタ。
Both	3	条件フィルタと値フィルタを組み合わせたフィルタ。

Operator 列挙体

ファイル `wijmo.grid.filter.js`
モジュール **wijmo.grid.filter**

フィルタ条件の演算子を指定します。

メンバー

名前	値	説明
EQ	0	等しい。
NE	1	等しくない。
GT	2	より大きい。
GE	3	以上。
LT	4	より小さい。
LE	5	以下。
BW	6	で始まる。
EW	7	で終わる。
CT	8	含む。
NC	9	含まない。

wijmo.grid.grouppanel モジュール

ファイル `wijmo.grid.grouppanel.js`
モジュール **wijmo.grid.grouppanel**

バインドされた**FlexGrid** コントロールでドラッグ＆ドロップによってグループを編集できるUIを提供する拡張です。

クラス

- GroupPanel

GroupPanel クラス

ファイル	wijmo.grid.grouppanel.js
モジュール	**wijmo.grid.grouppanel**
基本クラス	**Control**
派生クラス	**WjGroupPanel**
表示	☐継承されたメンバー ☐イベント発生元

GroupPanel コントロールは、バインドされた**FlexGrid** コントロールでドラッグ＆ドロップによってグループを編集できるUIを提供します。

ユーザーは**FlexGrid** の列をパネルにドラッグしてグループを作成し、 パネル内でグループを移動してグループ化の順序を変更できます。また、パネルのグループマーカーをクリックして、 グループ列を基準にソートしたり、グループを削除したりすることもできます。

GroupPanel を使用するには、**FlexGrid** コントロールを含むページにGroupPanelを追加し、 パネルの**grid** プロパティを**FlexGrid** コントロールに設定します。例:

```
// FlexGridを作成します。
var flex = new wijmo.grid.FlexGrid('#flex-grid');
flex.itemsSource = getData();

// データグループを編集するためのGroupPanelを追加します。
var groupPanel = new wijmo.grid.grouppanel.GroupPanel('#group-panel');
groupPanel.placeholder = "Drag columns here to create groups.";
groupPanel.grid = flex;
```

次の例では、**GroupPanel** コントロールを使用してOutlookスタイルのグループを **FlexGrid** コントロールに追加する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/hf7ud7ge)

The **GroupPanel** control relies on the grouping functionality provided by the **CollectionView** class, and therefore can be used only with bound grids.

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filter
- grid
- hideGroupedColumns
- hostElement
- isDisabled
- isTouching
- isUpdating
- maxGroups
- placeholder
- rightToLeft

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, options?): GroupPanel
```

GroupPanel クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **GroupPanel**

プロパティ

STATIC controlTemplate

GroupPanel コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

filter

Gets or sets the **FlexGridFilter** to use for filtering the grid data.

If you set this property to a valid filter, the group descriptors will display filter icons that can be used to see and edit the filer conditions associated with the groups.

型 **FlexGridFilter**

grid

この**GroupPanel** に接続している**FlexGrid** を取得または設定します。

グリッドをパネルに接続すると、 グリッドのデータソースで定義されているグループがパネルに表示されます。ユーザーはグリッド列をパネルにドラッグして新しいグループを作成できます。 また、パネル内でグループをドラッグしてグループ化の順序を変更したり、 パネルの項目を削除することによってグループを削除することもできます。

型 **FlexGrid**

hideGroupedColumns

グループ化列をオーナーグリッドで非表示にするかどうかを示す値を取得または設定します。

FlexGrid では行ヘッダにグループ化情報が表示されるので、 グループ化列を表示したままにすると情報が重複するため、 通常はグループ化列を非表示にすることを推奨します。

The default value for this property is **true**.

型 **boolean**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- maxGroups

許可されるグループの最大数を取得または設定します。 Setting this property to -1 allows any number of groups to be created. Setting it to zero prevents any grouping.

The default value for this property is **6**.

型 **number**

- placeholder

グループがないときにコントロールに表示する文字列を取得または設定します。 The default value for this property is '' (empty string).

型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(): void
```

パネルを更新して現在のグループを表示します。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

wijmo.grid.detail モジュール

ファイル `wijmo.grid.detail.js`
モジュール **wijmo.grid.detail**

FlexGrid コントロールの詳細行を提供する拡張機能。

クラス

- DetailMergeManager
- DetailRow
- FlexGridDetailProvider

列挙体

- DetailVisibilityMode
- KeyAction

DetailMergeManager クラス

ファイル wijmo.grid.detail.js
モジュール **wijmo.grid.detail**
基本クラス **MergeManager**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGridDetailProvider クラスによって使用される結合マネージャクラス。

DetailMergeManager は、詳細セル（**DetailRow** 内のセル）を すべてのグリッド列にまたがる1つの詳細セルにマージします。

コンストラクタ

- constructor

メソッド

- getMergedRange

コンストラクタ

constructor

```
constructor(grid: FlexGrid): DetailMergeManager
```

DetailMergeManager クラスの新しいインスタンスを初期化します。

パラメーター

- **grid**: **FlexGrid**
  この**DetailMergeManager** を所有する**FlexGrid** オブジェクト。

戻り値 **DetailMergeManager**

メソッド

- getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

GridPanel 内のセルの結合範囲を示す**CellRange** を取得します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。
- **r**: **number**
  セルが含まれる行のインデックス。
- **c**: **number**
  セルが含まれる列のインデックス。
- **clip**: **boolean** OPTIONAL
  結合範囲をグリッドの現在のビュー範囲にクリップするかどうか。

戻り値 **CellRange**

DetailRow クラス

ファイル wijmo.grid.detail.js
モジュール **wijmo.grid.detail**
基本クラス **Row**
表示 ☐ 継承されたメンバー ☐ イベント発生元

すべてのグリッド列にまたがる1つの詳細セルを含む行。

コンストラクタ

- constructor

プロパティ

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- dataItem
- detail
- grid
- height
- index
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- multiLine
- pos
- renderHeight
- renderSize
- size
- visible
- visibleIndex
- wordWrap

メソッド

- onPropertyChanged

コンストラクタ

constructor

```
constructor(parentRow: Row): DetailRow
```

DetailRow クラスの新しいインスタンスを初期化します。

パラメーター

- **parentRow**: **Row**
  この**DetailRow** によって詳細が提供される**Row**。

戻り値 **DetailRow**

プロパティ

allowDragging

ユーザーがマウスで行または列を新しい位置に移動できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

allowMerging

行または列にあるセルを結合できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

allowResizing

ユーザーがマウスで行または列をサイズ変更できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

collectionView

この行または列にバインドされた**ICollectionView** を取得します。

継承元 **RowCol**
型 **ICollectionView**

cssClass

行または列のヘッダ以外のセルをレンダリングするときに使用するCSSクラス名を取得または設定します。

継承元 **RowCol**
型 **string**

dataItem

項目がバインドされているデータコレクション内の項目を取得または設定します。

継承元 **Row**
型 **any**

- detail

このDetailRow 内の詳細セルを表すHTML要素を取得または設定します。

型 **HTMLElement**

- grid

行または列を所有するFlexGrid を取得します。

継承元 **RowCol**
型 **FlexGrid**

- height

行または列の高さを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

継承元 **Row**
型 **number**

- index

行または列の親コレクション内でのインデックスを取得します。

継承元 **RowCol**
型 **number**

- isContentHtml

この行または列にあるセルがプレーンテキストではなくHTMLコンテンツを含むかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isReadOnly

行または列のセルを編集できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isSelected

行または列が選択されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isVisible

行または列が表示可能で、なおかつ折りたたまれていないかどうかを示す値を取得または設定します。

このプロパティは読み取り専用です。行または列の表示/非表示設定を変更するには、代わりにvisible プロパティを使用してください。

継承元 **RowCol**
型 **boolean**

- multiLine

この行または列にあるセルの内容が改行文字(\n).でラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- pos

行または列の位置を取得します。

継承元 **RowCol**
型 **number**

- renderHeight

行のレンダリング高さを取得します。

列の表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮した幅が返されます。

継承元 **Row**
型 **number**

- renderSize

行または列のレンダリングサイズを取得します。 表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮したサイズが返されます。

継承元 **RowCol**
型 **number**

- size

行または列のサイズを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

継承元 **RowCol**
型 **number**

- visible

行または列が表示されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- visibleIndex

非表示にした要素(**`isVisible`**)を無視し、親コレクション内の行または列のインデックスを取得します。

継承元 **RowCol**
型 **number**

wordWrap

この行または列のセルの内容を使用可能な列幅に収まれるようにラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

メソッド

onPropertyChanged

```
onPropertyChanged(): void
```

オーナーリストをダーティとしてマークし、オーナーグリッドを更新します。

継承元 **RowCol**
戻り値 **void**

FlexGridDetailProvider クラス

ファイル wijmo.grid.detail.js
モジュール **wijmo.grid.detail**
派生クラス **WjFlexGridDetail**

FlexGrid コントロールの詳細行を実装します。

FlexGrid コントロールに詳細行を追加するには、 **FlexGridDetailProvider** のインスタンスを作成し、詳細セルに 表示される要素を作成する関数を **createDetailCell** プロパティで設定します。

次に例を示します。

```
// カテゴリを表示するためのFlexGridを作成します
var gridCat = new wijmo.grid.FlexGrid('#gridCat');
gridCat.itemsSource = getCategories();

// カテゴリごとの製品を表示する詳細行を追加します
var detailProvider = new wijmo.grid.detail.FlexGridDetailProvider(gridCat);
detailProvider.createDetailCell = function (row) {
  var cell = document.createElement('div');
  var gridProducts = new wijmo.grid.FlexGrid(cell);
  gridProducts.itemsSource = getProducts(row.dataItem.CategoryID);
  return cell;
}
```

FlexGridDetailProvider には、詳細行をいつ表示するかを決定する **detailVisibilityMode** プロパティがあります。このプロパティのデフォルト値は **ExpandSingle**です。これは、行ヘッダーに折りたたみ/展開アイコン を追加します。

次の例では、**FlexGridDetailProvider** を使用して**FlexGrid** の行にさまざまなタイプの詳細を追加する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/r9up8exz)

コンストラクタ

- constructor

プロパティ

- createDetailCell
- detailVisibilityMode
- disposeDetailCell
- grid
- isAnimated
- keyActionEnter
- maxHeight
- rowHasDetail

メソッド

- getDetailRow
- hideDetail
- isDetailAvailable
- isDetailVisible
- showDetail

コンストラクタ

constructor

```
constructor(grid: FlexGrid, options?: any): FlexGridDetailProvider
```

FlexGridDetailProvider クラスの新しいインスタンスを初期化します。

パラメーター

- **grid**: **FlexGrid**
  詳細行を受け取る**FlexGrid**。
- **options**: **any** OPTIONAL
  新しい**FlexGridDetailProvider** の初期化オプション。

戻り値 **FlexGridDetailProvider**

プロパティ

createDetailCell

詳細セルを作成するためのコールバック関数を取得または設定します。

このコールバック関数は、パラメータとして **Row** を受け取り、 行詳細を表すHTML要素を返します。 次に例を示します。

```
// 指定された行の詳細セルを作成します
dp.createDetailCell = function (row) {
  var cell = document.createElement('div');
  var detailGrid = new wijmo.grid.FlexGrid(cell, {
    itemsSource: getProducts(row.dataItem.CategoryID),
    headersVisibility: wijmo.grid.HeadersVisibility.Column
  });
  return cell;
};
```

型 **Function**

detailVisibilityMode

行詳細をいつ表示するかを決定する値を取得または設定します。 The default value for this property is **DetailVisibilityMode.ExpandSingle**.

型 **DetailVisibilityMode**

disposeDetailCell

詳細セルを破棄するためのコールバック関数を取得または設定します。

このコールバック関数は、パラメータとして **Row** を受け取り、 詳細セルに関連付けられているすべてのリソースを破棄します。

この関数はオプションです。この関数は、自動的に ガベージコレクトされないリソースを **createDetailCell** 関数で 割り当てている場合に使用します。

型 **Function**

grid

この**FlexGridDetailProvider** を所有する**FlexGrid**。

型 **FlexGrid**

isAnimated

行詳細を表示するときにアニメーションを使用するかどうかを示す値を取得または設定します。 The default value for this property is **false**.

型 **boolean**

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティのデフォルト設定は**`None`** です。これにより、グリッドはキーを処理できます。 そして**`ToggleDetail`** により、Enterキーを操作して詳細行の表示を切り替えます。

型 **`KeyAction`**

maxHeight

詳細行の最大高さをピクセル単位で取得または設定します。 The default value for this property is **null**, which means there's no upper limit to the detail row height.

型 **number**

rowHasDetail

行に詳細があるかどうかを判定するためのコールバック関数を取得または設定します。

このコールバック関数は、パラメータとして **`Row`** を受け取り、 行に詳細があるかどうかを示すブール値を返します。次に例を示します。

```
// 奇数のCategoryIDを持つ項目から詳細を削除します
dp.rowHasDetail = function (row) {
  return row.dataItem.CategoryID % 2 == 0;
};
```

このプロパティをnullに設定すると、すべての行に詳細が含まれます。

型 **Function**

メソッド

getDetailRow

```
getDetailRow(row: any): DetailRow
```

指定したグリッド行に関連付けられた詳細行を取得します。

パラメーター

- **row**: **any**
  調査する行または行のインデックス。

戻り値 **`DetailRow`**

hideDetail

```
hideDetail(row?: any): void
```

指定された行の詳細行を非表示にします。

パラメーター

- **row**: **any** OPTIONAL
  詳細が非表示になっている行または行のインデックス。このパラメータはオプションです。 指定されない場合は、すべての詳細行が非表示になります。

戻り値 **void**

isDetailAvailable

```
isDetailAvailable(row: any): boolean
```

行に表示する詳細があるかどうかを決定する値を取得します。

パラメーター

- **row**: **any**
  調査する行または行のインデックス。

戻り値 **boolean**

isDetailVisible

```
isDetailVisible(row: any): boolean
```

行の詳細を表示するかどうかを決定する値を取得します。

パラメーター

- **row**: **any**
  調査する行または行のインデックス。

戻り値 **boolean**

showDetail

```
showDetail(row: any, hideOthers?: boolean): void
```

指定された行の詳細行を表示します。

パラメーター

- **row**: **any**
  詳細が表示されている行または行のインデックス。

- **hideOthers**: **boolean** OPTIONAL
  他のすべての行の詳細を非表示にするかどうか。

戻り値 **void**

DetailVisibilityMode 列挙体

ファイル `wijmo.grid.detail.js`
モジュール **wijmo.grid.detail**

行詳細がいつどのように表示されるかを指定します。

メンバー

名前	値	説明
Code	0	詳細は、コードで表示/非表示にされます。それには、 **showDetail** メソッドと **hideDetail** メソッドを使用します。
Selection	1	詳細は、現在選択されている行に対して表示されます。
ExpandSingle	2	詳細は、行ヘッダーに追加されたボタンを使用して表示/非表示にされます。一度に1行のみを展開できます。
ExpandMulti	3	詳細は、行ヘッダーに追加されたボタンを使用して表示/非表示にされます。一度に複数行を展開できます。

KeyAction 列挙体

ファイル `wijmo.grid.detail.js`
モジュール **wijmo.grid.detail**

［ENTER］キーが押されたときに実行されるアクションを定義する定数を指定します。

メンバー

名前	値	説明
None	0	特別なアクションの必要はありません（グリッドがキーを処理します）。
ToggleDetail	1	詳細行の表示を切り替えます。

wijmo.grid.xlsx モジュール

ファイル `wijmo.grid.xlsx.js`
モジュール **wijmo.grid.xlsx**

FlexGrid コントロールにクライアント側でのExcel xlsxファイルの保存/ロード機能を提供する**FlexGridXlsxConverter** クラスを定義する拡張機能。

クラス

- FlexGridXlsxConverter
- XlsxFormatItemEventArgs

インターフェイス

- IExtendedSheetInfo
- IFlexGridXlsxOptions

FlexGridXlsxConverter クラス

ファイル `wijmo.grid.xlsx.js`
モジュール **wijmo.grid.xlsx**

このクラスは、**FlexGrid** コントロール　がExcel xlsxファイルからのロード/保存を行うための静的な**load**および**save**メソッドを提供します。

次の例では、**FlexGridXlsxConverter** を使用して**FlexGrid** コントロールの内容をXLSXにエクスポートする方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/yzcefvLy)

メソッド

- load
- loadAsync
- save
- saveAsync

メソッド

STATIC load

```
load(grid: FlexGrid, workbook: any, options?: IFlexGridXlsxOptions): void
```

xlsxファイルのコンテンツを含む**Workbook** インスタンスまたはBlobオブジェクトを**FlexGrid** インスタンスにロードします。 このメソッドは、JSZip 2.5で動作します。

次に例を示します。

```
// このサンプルは、［ファイルを開く］ダイアログで選択したxlsxファイルを開き、
// FlexGridに最初のシートの内容を挿入します。


// HTML
<input type="file"
    id="importFile"
    accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
/>
<div id="flexHost"></>


// JavaScript
var flexGrid = new wijmo.grid.FlexGrid("#flexHost"),
    importFile = document.getElementById('importFile');

importFile.addEventListener('change', function () {
    loadWorkbook();
});

function loadWorkbook() {
    var reader,
        file = importFile.files[0];
    if (file) {
        reader = new FileReader();
        reader.onload = function (e) {
            wijmo.grid.xlsx.FlexGridXlsxConverter.load(flexGrid, reader.result,
                { includeColumnHeaders: true });
        };
        reader.readAsArrayBuffer(file);
    }
}
```

パラメーター

- **grid**: **FlexGrid**
  Workbook オブジェクトをロードする**FlexGrid**。
- **workbook**: **any**
  xlsxファイルコンテンツを含む**Workbook**、 Blob、base-64文字列、またはArrayBuffer。
- **options**: **IFlexGridXlsxOptions** OPTIONAL
  ロードオプションを指定する**IFlexGridXlsxOptions** オブジェクト。

戻り値 **void**

STATIC loadAsync

```
loadAsync(grid: FlexGrid, workbook: any, options?: IFlexGridXlsxOptions, onLoaded?: Workbook), onError?: (reason?: any)): void
```

Workbook またはxlsxファイルを表すBlobを**FlexGrid** に非同期にロードします。

このメソッドにはJSZip 3.0が必要です。

パラメーター

- **grid**: **FlexGrid**
  Workbook オブジェクトをロードする**FlexGrid**。
- **workbook**: **any**
  xlsxファイルのコンテンツを表す**Workbook**、Blob、base64文字列、またはArrayBuffer。
- **options**: **IFlexGridXlsxOptions** OPTIONAL
  ロードオプションを指定する**IFlexGridXlsxOptions** オブジェクト。
- **onLoaded**: **(workbook: wijmo.xlsx.Workbook)** OPTIONAL
  メソッドの実行が終了すると呼び出されるコールバック。 このコールバックは、ロードされたワークブックへのアクセスを提供します。 これは、パラメータとしてコールバックに渡されます。
- **onError**: **(reason?: any)** OPTIONAL
  ファイル保存時にエラーがあると呼び出されるコールバック。 これは、パラメータとしてコールバックに渡されます。

  次に例を示します。

```
wijmo.grid.xlsx.FlexGridXlsxConverter.loadAsync(grid, blob, null, function (workbook) {

     // このコールバックで、ユーザーはロードされたワークブックインスタンスにアクセスできます。
     var app = worksheet.application ;
     ...
}, function (reason) {

     // このコールバックで、ユーザーは失敗理由を確認できます。
     console.log('The reason of save failure is ' + reason);
});
```

戻り値 **void**

STATIC save

```
save(grid: FlexGrid, options?: IFlexGridXlsxOptions, fileName?: string): Workbook
```

FlexGrid インスタンスを**Workbook** インスタンスに保存します。 このメソッドは、JSZip 2.5で動作します。

次に例を示します。

```
// このサンプルは、ボタンのクリックで、FlexGridのコンテンツをxlsxファイルにエクスポートします。
// click.


// HTML
<button
    onclick="saveXlsx('FlexSheet.xlsx')">
    Save
</button>


// JavaScript
function saveXlsx(fileName) {

    // flexGridをxlsxファイルに保存します。
    wijmo.grid.xlsx.FlexGridXlsxConverter.save(flexGrid,
            { includeColumnHeaders: true }, fileName);
}
```

パラメーター

- **grid**: **FlexGrid**
  保存されるFlexGrid。
- **options**: **IFlexGridXlsxOptions** OPTIONAL
  保存オプションを指定する**IFlexGridXlsxOptions** オブジェクト。
- **fileName**: **string** OPTIONAL
  生成されるファイル名。

戻り値 **Workbook**

STATIC saveAsync

```
saveAsync(grid: FlexGrid, options?: IFlexGridXlsxOptions, fileName?: string, onSaved?: (base64: string), onError?: (reason?: any)): Workbook
```

FlexGrid のコンテンツをファイルに非同期に保存します。

このメソッドにはJSZip 3.0が必要です。

パラメーター

- **grid**: **FlexGrid**
  保存されるFlexGrid。

- **options**: **IFlexGridXlsxOptions** OPTIONAL
  保存オプションを指定する**IFlexGridXlsxOptions** オブジェクト。

- **fileName**: **string** OPTIONAL
  生成されるファイル名。

- **onSaved**: **(base64: string)** OPTIONAL
  メソッドの実行が終了すると呼び出されるコールバック。 このコールバックは、保存されたワークブックのコンテンツへのアクセスを提供します。 これは、base64文字列としてエンコードされ、パラメータとしてコールバックに渡されます。

- **onError**: **(reason?: any)** OPTIONAL
  ファイル保存時にエラーがあると呼び出されるコールバック。 これは、パラメータとしてコールバックに渡されます。

  次に例を示します。

```
wijmo.grid.xlsx.FlexGridXlsxConverter.save(flexGrid,
    { includeColumnHeaders: true }, // オプション
    'FlexGrid.xlsx', // ファイル名
    function (base64) { // onSaved
        // このコールバックで、ユーザーはbase64文字列にアクセスできます。
        document.getElementById('export').href = 'data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;' + 'base64,' + base64;
    },
    function (reason) { // onError
        // このコールバックで、ユーザーは失敗理由を確認できます。
        console.log('The reason of save failure is ' + reason);
    }
);
```

戻り値 **Workbook**

XlsxFormatItemEventArgs クラス

ファイル wijmo.grid.xlsx.js
モジュール **wijmo.grid.xlsx**
基本クラス **CellRangeEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

IFlexGridXlsxOptions.formatItemコールバックの引数を表します。

コンストラクタ

- constructor

プロパティ

- cancel
- cell
- col
- data
- empty
- panel
- range
- row
- xlsxCell

メソッド

- getFormattedCell

コンストラクタ

constructor

```
constructor(p: GridPanel, rng: CellRange, data?: any): CellRangeEventArgs
```

CellRangeEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。
- **rng**: **CellRange**
  イベントの影響を受けるセルの範囲。
- **data**: **any** OPTIONAL
  このイベントに関連するデータ。

継承元 **CellRangeEventArgs**
戻り値 **CellRangeEventArgs**

プロパティ

cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

cell

IFlexGridXlsxOptions.includeCellStylesがtrueに設定されている場合は、 書式設定されたグリッドセルを表す要素への参照を含みます。そうでない場合は、null値です。

型 **HTMLElement**

col

このイベントの影響を受けた列を取得します。

継承元 **CellRangeEventArgs**
型 **number**

data

このイベントに関連するデータを取得または設定します。

継承元 **CellRangeEventArgs**
型 **any**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

panel

このイベントの影響を受ける **GridPanel** を取得します。

継承元 **CellRangeEventArgs**
型 **GridPanel**

range

このイベントの影響を受ける **CellRange** を取得します。

継承元 **CellRangeEventArgs**
型 **CellRange**

row

このイベントの影響を受けた行を取得します。

継承元 **CellRangeEventArgs**
型 **number**

- xlsxCell

エクスポートするセル表現を格納します。初期状態では、FlexGridエクスポートによって作成されたデフォルトのセル表現を格納し、 イベントハンドラによって変更することで、最終的な内容をカスタマイズすることができます。たとえば、xlsxCell.valueプロパティでセルの内容の変更、xlsxCell.styleプロパティでセルのスタイルの変更などを行うことができます。

型 **IWorkbookCell**

メソッド

- getFormattedCell

```
getFormattedCell(): HTMLElement
```

カスタム書式(formatItemイベント、セルテンプレート)が適用されたセルを返します。 このメソッドは、カスタム書式のエクスポートが無効(IFlexGridXlsxOptions.includeCellStyles=false)だが、いくつかのセルのカスタムコンテンツ、スタイル、またはその両方を エクスポートする必要がある場合に便利です。

戻り値 **HTMLElement**

IExtendedSheetInfo インターフェイス

ファイル wijmo.grid.xlsx.js
モジュール **wijmo.grid.xlsx**

FlexGrid インスタンスの動的な**wj_sheetInfo**プロパティを介してアクセスできる追加のワークシートプロパティを定義します。

プロパティ

- evaluateFormula
- fonts
- mergedRanges
- name
- styledCells
- tables
- visible

プロパティ

- evaluateFormula

A function that evaluates the formula of cell.

型 **Function**

- fonts

シートで使用されているフォント名の配列が含まれます。

型 **string**[]

- mergedRanges

シート内の結合された範囲。

型 **CellRange**[]

- name

シート名。

型 **string**

- styledCells

シート内のスタイル設定されたセル。

型 **any**

- tables

このワークシート内のテーブル。

型 **WorkbookTable**[]

visible

シートの表示/非表示設定。

型 **boolean**

IFlexGridXlsxOptions インターフェイス

ファイル wijmo.grid.xlsx.js
モジュール **wijmo.grid.xlsx**

FlexGrid Xlsx変換オプション

プロパティ

- activeWorksheet
- formatItem
- includeCellStyles
- includeColumnHeaders
- includeColumns
- includeRowHeaders
- sheetIndex
- sheetName
- sheetVisible

プロパティ

activeWorksheet

xlsxファイル内のアクティブなシートのインデックスまたは名前。

型 **any**

formatItem

エクスポートされたセルごとに呼び出されるオプションのコールバック。エクスポートされたセルの値およびスタイルの 変換を行うことができます。コールバックは、includeCellStylesプロパティの値にかかわらず呼び出されます。 ソースグリッドセルに関する情報とエクスポートされるファイルでの表現を定義する**`IWorkbookCell`** オブジェクトを提供する **`XlsxFormatItemEventArgs`** 型の単一のパラメータがあります。

型

includeCellStyles

生成されるxlsxファイルにセルのスタイル設定を含めるかどうかを示します。

型 **boolean**

includeColumnHeaders

生成されるxlsxファイルの最初の行に列ヘッダーを表示するかどうかを示します。

型 **boolean**

- includeColumns

エクスポートに入れる、または除外する必要があるFlexGridの列を示すコールバック。

例:

```
// このサンプルコードでは、'country'列をエクスポートから除外します。
```

```
// JavaScript
wijmo.grid.xlsx.FlexGridXlsxConverter.save(grid, {
  includeColumns: function(column) {
     return column.binding !== 'country';
  }
}
```

型

- includeRowHeaders

生成されるxlsxファイルの最初の行に列ヘッダーを表示するかどうかを示します。

型 **boolean**

- sheetIndex

ワークブック内のシートのインデックス。どのシートをインポートするかを示します。

型 **number**

- sheetName

シートの名前。 インポートする際、どのシートをインポートするかを示します。sheetIndexとsheetNameの両方が設定されている場合、sheetNameがsheetIndexより優先されます。 エクスポートするワークシートの名前を設定します。

型 **string**

- sheetVisible

シートの表示状態。

型 **boolean**

wijmo.grid.multirow モジュール

ファイル `wijmo.grid.multirow.js`
モジュール **wijmo.grid.multirow**

MultiRow コントロールとそのコントロールに関連付けられたクラスを定義します。

クラス

- MultiRow

MultiRow クラス

ファイル wijmo.grid.multirow.js
モジュール **wijmo.grid.multirow**
基本クラス **FlexGrid**
派生クラス **WjMultiRow**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid コントロールを拡張して、項目ごとに複数の行を提供します。

layoutDefinitionプロパティを使用して、各データ項目の表示に使用する 行のレイアウトを定義します。

MultiRow コントロールでは、カスタム複数行レイアウトと干渉するいくつかの **FlexGrid** プロパティが無効にされています。 無効にされたプロパティには、**allowMerging** や **childItemsPath** があります。

コンストラクタ

- constructor

プロパティ

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoScroll
- autoSearch
- autoSizeMode
- bottomLeftCells
- cellFactory
- cells
- centerHeadersVertically
- childItemsPath
- clientSize
- cloneFrozenCells
- collapsedHeaders
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- deferResizing
- editableCollectionView
- editRange
- frozenColumns
- frozenRows
- groupHeaderFormat

- headersVisibility
- hostElement
- imeEnabled
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemValidator
- keyActionEnter
- keyActionTab
- layoutDefinition
- mergeManager
- newRowAtTop
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- rightToLeft
- rowHeaderPath
- rowHeaders
- rows
- rowsPerItem
- scrollPosition
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionMode
- showAlternatingRows
- showDropDown
- showErrors
- showGroups
- showHeaderCollapseButton
- showMarquee
- showSelectedHeaders
- showSort
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- validateEdits
- viewRange
- virtualizationThreshold

メソッド

- addEventListener
- applyTemplate
- autoSizeColumn

- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseGroupsToLevel
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getBindingColumn
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCollapsedHeadersChanged
- onCollapsedHeadersChanging
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged

- onGroupCollapsedChanging
- onItemsSourceChanged
- onItemsSourceChanging
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onRefreshed
- onRefreshing
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- startEditing
- toggleDropDownList

イベント

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit

- cellEditEnded
- cellEditEnding
- collapsedHeadersChanged
- collapsedHeadersChanging
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- itemsSourceChanging
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- refreshed
- refreshing
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged
- selectionChanging
- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout
- updatingView

コンストラクタ

constructor

```
constructor(element: any, options?): MultiRow
```

MultiRow クラスの新しいインスタンスを初期化します。

通常、**options**パラメータには**layoutDefinition** プロパティの 値が含まれます。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **MultiRow**

プロパティ

activeEditor

現在アクティブになっているセルエディタを表す**HTMLInputElement**を取得します。

継承元 **FlexGrid**
型 **HTMLInputElement**

allowAddNew

ユーザーがソースコレクションに項目を追加できるようにグリッドに新規行テンプレートを表示するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、新規行テンプレートは表示されません。

継承元 **FlexGrid**
型 **boolean**

allowDelete

ユーザーが［Delete］キーを押したときにグリッドの選択されている行を削除するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、選択されている行は削除されません。

継承元 **FlexGrid**
型 **boolean**

allowDragging

ユーザーがマウスを使用して行、列、またはその両方をドラッグできるかどうかを決定する値を取得または設定します。

autoScroll プロパティがtrueに設定されている場合、ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされます。

グリッドでは、列のドラッグがデフォルトで有効になっています。

グリッドが連結モードでは、行をドラッグするには特別な 考慮が必要になります。

グリッドが連結モードでは、行をドラッグするとデータソースとの同期が失われます（たとえば行4は6行として参照されることがあります）。 これを回避するには、**draggedRow** イベントを処理し、データを新しい行の位置と同期させる必要があります。

また、**allowSorting** プロパティをfalseに設定する必要があります。そうしないと、行の順序がデータによって決定され、行をドラッグするのは無意味になります。

次のフィドルでは、連結グリッドでの行のドラッグを実行しています。 **Bound Row Dragging (http://jsfiddle.net/Wijmo5/kyg0qsda/)**.

継承元 **FlexGrid**
型 **AllowDragging**

allowMerging

グリッドのどの部分のセル結合を許可するかを取得または設定します。

継承元 **FlexGrid**
型 **AllowMerging**

allowResizing

ユーザーがマウスを使用して行、列、またはその両方をサイズ変更できるかどうかを決定する値を取得または設定します。

サイズ変更が可能な場合は、列ヘッダーセルの右端をドラッグして列を、行ヘッダーセルの下端をドラッグして行をサイズ変更できます。

ヘッダーセルの端をダブルクリックして、行および列をコンテンツに合わせて自動的にサイズ変更することもできます。 自動サイズ変更の動作は、**autoSizeMode** プロパティを使用してカスタマイズできます。

継承元 **FlexGrid**
型 **AllowResizing**

allowSorting

ユーザーが列ヘッダーセルをクリックして列をソートできるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

autoClipboard

グリッドがクリップボードショートカットを処理するかどうかを決定する値を取得または設定します。

次のクリップボードショートカットがあります。

［Ctrl］+［C］、［Ctrl］+［Ins］	グリッドの選択範囲をクリップボードにコピーします。
［Ctrl］+［V］、［Shift］+［Ins］	クリップボードのテキストをグリッドの選択範囲に貼り付けます。

クリップボード操作は、表示されている行および列だけが対象となります。

読み取り専用のセルは、貼り付け操作の影響を受けません。

継承元 **FlexGrid**
型 **boolean**

autoGenerateColumns

グリッドが**itemsSource** に基づいて列を自動的に生成するかどうかを 決定する値を取得または設定します。

列の生成は、少なくとも1項目を含む**itemsSource** プロパティに依存します。このデータ項目が検査され、列が作成されて、プリミティブ値（数値、文字列、Boolean、またはDate）を含むプロパティに連結されます。

プロパティをnullに設定すると、適切なタイプを推測する方法がないため、列は生成されません。このような場合は、**autoGenerateColumns** プロパティをfalseに設定し、明示的に列を作成する必要があります。次に例を示します。

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // データ項目にnull値が含まれる場合があります
  columns: [                  // そのため、列を明示的に定義します
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

継承元 **FlexGrid**
型 **boolean**

autoScroll

ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされるかどうかを決定する値を取得または設定します。

行と列のドラッグは、**allowDragging** プロパティによって制御されます。

このプロパティはデフォルトでtrueに設定されます。

継承元 **FlexGrid**
型 **boolean**

autoSearch

ユーザーが読み取り専用セルに入力するに伴ってグリッドでセルを検索するかどうかを決定する値を取得または設定します。

検索が現在選択されている列(編集不可能な場合)で行われます。検索は現在選択されている行から始まり、大文字と小文字を区別されません。

継承元 **FlexGrid**
型 **boolean**

autoSizeMode

行または列のサイズを自動設定するときにどのセルを考慮に入れるかを取得または設定します。

このプロパティは、ユーザーが列ヘッダの端をダブルクリックしたときの動作を制御します。

デフォルトでは、その列のヘッダセルとデータセルの内容に基づいて列の幅が自動的に設定されます。このプロパティを使用すると、ヘッダのみまたはデータのみに基づいて列の幅を設定するように変更できます。

継承元 **FlexGrid**
型 **AutoSizeMode**

bottomLeftCells

左下のセルを含む**GridPanel** を取得します。

bottomLeftCells パネルは、行ヘッダの下、**columnFooters** パネルの左に表示されます。

継承元 **FlexGrid**
型 **GridPanel**

cellFactory

このグリッドのセルを作成および更新する**CellFactory** を取得または設定します。

継承元 **FlexGrid**
型 **CellFactory**

cells

データセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

centerHeadersVertically

複数の行にまたがるセルのコンテンツを垂直方向に中央揃えにするかどうかを 判定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

childItemsPath

階層グリッドで子の行の生成に使用される 1 つ以上のプロパティの名前を取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `childItemsPath = 'items'` ）。

項目の異なるレベルに異なる名前を持つ子項目がある場合は、このプロパティに、各レベルの子項目を含むプロパティの名前から成る配列を設定します。（たとえば、 `childItemsPath = ['checks','earnings'];` ）。

サンプル

Show me (https://jsfiddle.net/Wijmo5/kk9j93bL)

継承元 **FlexGrid**
型 **any**

clientSize

コントロールのクライアントサイズを取得します（コントロールのサイズからヘッダーとスクロールバーを引いた値）。

継承元 **FlexGrid**
型 **Size**

cloneFrozenCells

スクロール中に知覚されるパフォーマンスを向上させるには、FlexGridがフリーズされたセルを複製し、別の要素に表示するかどうかを決定する値を取得または設定します。

このプロパティのデフォルト値はnullであり、ブラウザーによって一番適当な設定が選択されます。

継承元 **FlexGrid**
型 **boolean**

collapsedHeaders

列ヘッダーを折りたたみ、単一の行としてグループヘッダーを 表示するかどうかを判定する値を取得または設定します。

collapsedHeadersプロパティをtrueに設定した場合は、 各グループの **header**プロパティを設定して、ヘッダーが空に ならないようにしてください。

`collapsedHeaders` プロパティを**null**に設定すると、 グリッドにすべてのヘッダー情報（グループおよび列）が表示されます。 この場合、最初の行にはグループヘッダーが、残りの行には個々の 列ヘッダーが表示されます。

The default value for this property is **false**.

型 **boolean**

collectionView

グリッドデータを含む**ICollectionView** を取得します。

継承元 **FlexGrid**
型 **ICollectionView**

columnFooters

列フッターセルを保持する**GridPanel** を取得します。

columnFooters パネルは、グリッドセルの下、**bottomLeftCells** パネルの右に表示されます。これを使用して、グリッドデータの下にサマリー情報を表示できます。

以下の例では、**columnFooters** パネルに 1行を追加して、**aggregate** プロパティが設定された列に サマリーデータを表示する方法を示します。

```
function addFooterRow(flex) {

  // 集計を表示するGroupRowを作成します
  var row = new wijmo.grid.GroupRow();

  // 列フッターパネルに行を追加します
  flex.columnFooters.rows.push(row);

  // ヘッダーにシグマを表示します
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

継承元 **FlexGrid**
型 **GridPanel**

columnHeaders

列ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

columnLayout

現在の列レイアウトを定義するJSON文字列を取得または設定します。

列レイアウト文字列は、列とそのプロパティを含む配列を表します。これを使用して、ユーザーが定義した列レイアウトをセッションを越えて保持できます。また、ユーザーが列レイアウトを変更できるアプリケーションで元に戻す/やり直し機能を実装することもできます。

データマップはシリアル化できないため、列レイアウト文字列に**dataMap** プロパティは含まれていません。

継承元 **FlexGrid**
型 **string**

columns

グリッドの列コレクションを取得します。

継承元 **FlexGrid**
型 **ColumnCollection**

controlRect

コントロールの外接矩形（ページ座標単位）を取得します。

継承元 **FlexGrid**
型 **Rect**

STATIC controlTemplate

FlexGrid コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **FlexGrid**
型 **any**

deferResizing

ユーザーがマウスボタンを放すまで、行および列のサイズ変更を遅らせるかどうかを決定する値を取得または設定します。

デフォルトでは、**deferResizing** はfalseに設定されており、ユーザーがマウスをドラッグするに伴って行および列がサイズ変更されます。このプロパティをtrueに設定すると、グリッドにサイズ変更マーカーが表示され、ユーザーがマウスボタンを放したときに初めて行または列のサイズが変更されます。

継承元 **FlexGrid**
型 **boolean**

editableCollectionView

グリッドデータを含む**IEditableCollectionView** を取得します。

継承元 **FlexGrid**
型 **IEditableCollectionView**

editRange

現在編集中のセルを識別する**CellRange** を取得します。

継承元 **FlexGrid**
型 **CellRange**

frozenColumns

固定された列の数を取得または設定します。

固定された列は水平方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

frozenRows

静止行の数を取得または設定します。

固定された行は垂直方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

groupHeaderFormat

グループヘッダーコンテンツを作成するために使用される書式文字列を取得または設定します。

この文字列は、任意のテキストと次の置換文字列を含むことができます。

- **{name}**：グループ化されるプロパティの名前。
- **{value}**：グループ化されるプロパティの値。
- **{level}**：グループレベル。
- **{count}**：このグループ内にある項目の合計数。

列がグループ化プロパティに連結されている場合は、列ヘッダーを使用して パラメータを置換し、列の書式とデータマップを使用して `{value}` パラメータを計算します。 列がない場合は、代わりにグループ情報が使用されます。

グループプロパティに連結された非表示の列を追加して、グループヘッダーセルの書式設定をカスタマイズすることもできます。

このプロパティのデフォルト値は
`'{name}: <b>{value}</b>({count:n0} items)'` です。 これは、 `'Country: UK (12 items)'` や
`'Country: Japan (8 items)'` のようなグループヘッダーを作成します。

継承元 **FlexGrid**
型 **string**

headersVisibility

行ヘッダおよび列ヘッダが表示されるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

imeEnabled

編集モードでないときに、グリッドがIME（Input Method Editor）をサポートするかどうかを決定する値を取得または設定します。

このプロパティは、日本語、中国語、韓国語など、IMEサポートを必要とする言語のサイト/アプリケーションにのみ関係します。

継承元 **FlexGrid**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してセル値を変更できるかどうかを判定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

このグリッドのセルのカスタマイズに使用されるフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意のセルに任意の内容を追加できます。そのため、グリッドセルの外観と動作を非常に柔軟に制御することが可能です。

この関数は、セルを含む**GridPanel**、セルの行インデックスと列インデックス、セルを表すHTML要素の4つのパラメーターをとります。通常は、関数内でセル要素の**innerHTML**　プロパティを変更するようにします。

例:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == wijmo.grid.CellType.Cell) {

    // セルにスパークラインを描画します。
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

FlexGridはセルをリサイクルするため、**itemFormatter** によってセルのスタイル属性を変更する場合は、 セルの適切でないスタイル属性を事前にリセットする必要があります。例:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // カスタマイズする属性をリセットします。
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // このセルのcolorおよびbackgroundColor属性をカスタマイズします。
  ...
}
```

複数のクライアントがグリッドのレンダリングをカスタマイズできるようにする場合は（たとえば、ディレクティブや再利用可能なライブラリを作成するときなど）、代わりに**formatItem** イベントを使用することを検討してください。このイベントを使用すると、複数のクライアントがそれぞれ独自のハンドラをアタッチできます。

継承元 **FlexGrid**
型 **Function**

itemsSource

グリッドに表示される項目を含む配列または**ICollectionView** を取得または設定します。

継承元 **FlexGrid**
型 **any**

itemValidator

セルに有効なデータが含まれているかどうかを決定するバリデータ関数を取得または設定します。

指定された場合、バリデータ関数はセルの行インデックスと列インデックスを含む2つのパラメータをとり、エラーの説明を含む文字列を返す必要があります。

このプロパティは、バインドされていないグリッドを処理する場合に特に便利です。バインドされたグリッドは、代わりに**getError** プロパティを使用して検証できます。

この例では、セルのすぐ上のセルと同じデータがセルに含まれないようにする方法を示します。

```
// 上のセルは、現在のセルと同じ値が含まれていないことを確認します
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'これは重複した値です...'
    }
  }
  return null; // エラーなし
}
```

継承元 **FlexGrid**
型 **Function**

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は**MoveDown** となり、コントロールがカーソルを次の行に移動します。 この動作は標準的なExcel式の動作です。

継承元 **FlexGrid**
型 **KeyAction**

keyActionTab

［Tab］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は、**None** となります。これにより、Tabキーが押されたときにブラウザ上で次または前のコントロールを選択できます。これは、ページのアクセシビリティを向上させるために推奨される設定になります。

以前のバージョンでは、デフォルトは**Cycle** に設定されていました。これにより、コントロールがカーソルを、グリッドを横切って下部に移動しました。これは標準的なExcelの動作ですが、アクセシビリティには適していません。

また、**CycleOut** の設定が存在します。これにより、カーソルがセルを通じて移動（**Cycle** ）してから、最後または最初のセルが選択されたときにページの次/前のコントロールに移動します 。

継承元 **FlexGrid**
型 **KeyAction**

layoutDefinition

各データ項目の表示に使用される行のレイアウトを定義する配列を取得または設定します。

この配列には、次のプロパティを持つセルグループオブジェクトのリストが含まれます。

- **header**: グループヘッダー（ヘッダーが折りたたまれたときに表示される）
- **colspan**: グループがまたがるグリッド列の数
- **cells**: セルオブジェクトの配列。これらは、**colspan**プロパティで**Column** を拡張します。

`layoutDefinition` プロパティを設定すると、グリッドは各グループ内のセルを 次のようにスキャンします。

1. グリッドは、グループ自体のcolspanとグループ内の最も広いセルのスパンの 大きい方をグループのcolspanとします。
2. セルがグループ内の現在の行に収まる場合、そのセルは現在の行に追加されます。
3. 収まらない場合は、新しい行に追加されます。

すべてのグループが準備できたら、すべてのグループの最大のrowspanを1レコードの行数とし、 必要に応じて各グループに行を追加して高さを合わせます。

このスキームは単純で柔軟です。次に例を示します。

```
{ header: 'Group 1', cells: [{ binding: 'c1' }, { bnding: 'c2'}, { binding: 'c3' }]}
```

このグループのcolspanは1なので、各列に1つのセルが使用されます。結果は次のとおりです。

```
| C1 |
| C2 |
| C3 |
```

2列を含むグループを作成するには、グループの **colspan**プロパティを次のように設定します。

```
{ header: 'Group 1', colspan: 2, cells:[{ binding: 'c1' }, { binding: 'c2'}, { binding: 'c3' }]}
```

セルは次のように折り返されます。

```
| C1  | C2 |
| C3       |
```

最後のセルは2列にまたがります（グループ全体に拡大）。

グループではなく個々のセルにcolspanを指定することもできます。次に例を示します。

```
{ header: 'Group 1', cells: [{binding: 'c1', colspan: 2 }, { bnding: 'c2'}, { binding: 'c3' }]}
```

最初のセルのcolspanが2なので、結果は次のようになります。

```
| C1       |
| C2 |  C3 |
```

セルは**`Column`** クラスを拡張しているため、どのセルにも通常の **`Column`** プロパティのすべてを追加できます。 次に例を示します。

```
{ header: 'Group 1', cells: [
   { binding: 'c1', colspan: 2 },
   { bnding: 'c2'},
   { binding: 'c3', format: 'n0', required: false, etc... }}
]}
```

型 **any**[]

mergeManager

セルの結合方法を決定する**MergeManager** オブジェクトを取得または設定します。

継承元 **FlexGrid**
型 **MergeManager**

newRowAtTop

新しい行テンプレートをグリッドの上部に配置するか、下部に配置するかを示す値を取得または設定します。

newRowAtTop プロパティをtrueに設定した場合、新しい行テンプレートを常に表示したままにする場合は、**frozenRows** プロパティを 1に設定します。これにより、新しい行テンプレートは上部に固定され、ビューからスクロールオフされなくなります。

allowAddNew プロパティがtrueに設定されている場合、および**itemsSource** オブジェクトが新しい項目の追加をサポートしている場合にのみ、新しい行テンプレートが表示されます。

継承元 **FlexGrid**
型 **boolean**

preserveOutlineState

データがリフレッシュされたときに、グリッドがノードの展開/折りたたみ状態を保持するかどうかを決定する値を取得または設定します。

preserveOutlineState プロパティの実装は、JavaScriptの**Map** オブジェクトに基づいています。このオブジェクトはIE 9およびIE 10で利用できません。

継承元 **FlexGrid**
型 **boolean**

preserveSelectedState

データがリフレッシュされたときに、グリッドが行の選択状態を保持するかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

quickAutoSize

グリッドが列のサイズを自動調整するときに精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、自動サイズ変更の機能が無効になります。このプロパティをtrueに設定すると、各列の**quickAutoSize** プロパティの値に従って、その機能が有効になります。null（デフォルト値）に設定した場合、カスタムの**itemFormatter** を持たないグリッドの機能、または**itemFormatter** イベントにアタッチされたハンドラの機能が有効になります。

継承元 **FlexGrid**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rowHeaderPath

行ヘッダーセルを作成するために使用されるプロパティの名前を取得または設定します。

行ヘッダーセルは表示されないし、選択することもできません。アクセシビリティツールと組み合わせて使用することを意図しています。

継承元 **FlexGrid**
型 **string**

rowHeaders

行ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

rows

グリッドの行コレクションを取得します。

継承元 **FlexGrid**
型 **RowCollection**

rowsPerItem

各項目の表示に使用される行数を取得します。

この値は、**layoutDefinition**プロパティの値に基づいて 自動的に計算されます。

型 **number**

scrollPosition

グリッドのスクロールバーの値を表す**Point** を取得または設定します。

継承元 **FlexGrid**
型 **Point**

scrollSize

グリッド内容のサイズ（ピクセル単位）を取得します。

継承元 **FlexGrid**
型 **Size**

selectedItems

現在選択されているデータ項目を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selectedRows

現在選択されている行を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any[]**

selection

現在の選択を取得または設定します。

継承元 **FlexGrid**
型 **CellRange**

selectionMode

現在の選択モードを取得または設定します。

継承元 **FlexGrid**
型 **SelectionMode**

showAlternatingRows

グリッドで交互表示行のセルに'wj-alt'クラスを追加するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、CSSを変更しなくても、交互表示行スタイルを無効にできます。

継承元 **FlexGrid**
型 **boolean**

showDropDown

グリッドで、**showDropDown** プロパティがtrueに設定されている列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **FlexGrid**
型 **boolean**

showErrors

グリッドで、検証エラーがあるセルとエラーの説明を含むツールチップに'wj-state-invalid' クラスを追加するかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**itemValidator** または**getError** プロパティを使用して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

showGroups

データグループを区切るためにグリッドにグループ行を挿入するかどうかを決定する値を 取得または設定します。

データグループを作成するには、グリッドの **itemsSource** として使用される **ICollectionView** オブジェクトの **groupDescriptions** プロパティを変更します。

継承元 **FlexGrid**
型 **boolean**

showHeaderCollapseButton

グリッドで、行フィールドの列ヘッダーにソートインジケータを グリッドの列ヘッダーパネルに表示するかどうかを判定する値を 取得または設定します。

ボタンが表示されている場合は、ボタンをクリックすると、 **collapsedHeaders**プロパティの値が切り替わります。

The default value for this property is **false**.

型 **boolean**

showMarquee

現在の選択範囲の周囲にマーキー要素を表示するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

showSelectedHeaders

選択されているヘッダセルを示すためにクラス名を追加するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

showSort

グリッドで、列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

ソートは、グリッドの **itemsSource** として使用される **ICollectionView** オブジェクトの **sortDescriptions** プロパティによって制御されます。

継承元 **FlexGrid**
型 **boolean**

sortRowIndex

ソートインジケーターを表示する列ヘッダパネルの行のインデックスを取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、 **columnHeaders** パネルの最後の行がソート行になります。

継承元 **FlexGrid**
型 **number**

stickyHeaders

ユーザーがドキュメントをスクロールしたときに、列ヘッダーを表示したままにするかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

topLeftCells

左上のセル（列ヘッダーの左）を含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

treeIndent

異なるレベルの行グループをオフセットするインデントを取得または設定します。

継承元 **FlexGrid**
型 **number**

validateEdits

検証に失敗した編集をユーザーがコミットしようとしたときに、グリッドを編集モードのままにするかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**getError** メソッドを呼び出して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

viewRange

現在表示されているセルの範囲を取得します。

継承元 **FlexGrid**
型 **CellRange**

virtualizationThreshold

仮想化を有効にするために必要な最小行数、最小列数、またはその両方を取得または設定します。

このプロパティはデフォルトでゼロに設定されます。つまり、仮想化は常に有効ということです。これにより、バインディングパフォーマンスとメモリー必要量が向上しますが、スクロール時のパフォーマンス低下は犠牲になります。

グリッドの行数が少ない場合（約50〜100）、このプロパティを150などのやや高い値に設定することで、スクロールのパフォーマンスを向上させることができます。これにより、仮想化が無効になり連結処理が遅くなりますが、認識されるスクロールのパフォーマンスが向上する可能性があります。たとえば、以下のコードは、データソースに150を超える項目がある場合、グリッドがセルを仮想化します。

```
// 150を超える項目がある場合はグリッドを仮想化します
theGrid.virtualizationThreshold = 150;
```

このプロパティを200より大きい値に設定することは推奨されません。ロード処理の時間が長くなり、グリッドはすべての行のセルを作成しているときに数秒間フリーズするため、ページ上の要素数が多いためブラウザが遅くなります。

行と列に別々の仮想化しきい値を設定する場合は、**virtualizationThreshold** プロパティを2つの数値を含む配列に設定できます。この場合、最初の数値は行のしきい値として使用され、2番目の数値は列のしきい値として使用されます。たとえば、次のコードでは、グリッドは行を仮想化しますが、列を仮想化しません。

```
// 行（しきい値0）は仮想化しますが、列は仮想化しません（しきい値10,000）
theGrid.virtualizationThreshold = [0, 10000];
```

継承元 **FlexGrid**
型 **any**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

列をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **c**: **number**
  サイズ変更する列のインデックス。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

列の範囲をその内容がちょうど収まるようにサイズ変更します。

測定される行の範囲は常に、現在表示されているすべての行 + 現在表示されていない最大2,000行です。グリッドに大量のデータが含まれている場合には（たとえば、50,000行など）、すべての行を測定すると非常に時間がかかる可能性があるため、すべての行は測定されません。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstColumn**: **number** OPTIONAL
  サイズ変更する最初の列のインデックス（デフォルトは最初の列）。
- **lastColumn**: **number** OPTIONAL
  サイズ変更する最後の列のインデックス（デフォルトは最後の列）。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

行をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **r**: **number**
  サイズ変更する行のインデックス。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の列であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

行の範囲をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックス。
- **lastRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックスす。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の行であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

canEditCell

```
canEditCell(r: number, c: number): void
```

指定されたセルが編集可能かどうかを示す値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

継承元 **FlexGrid**
戻り値 **void**

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

すべてのグループ行を指定したレベルに折りたたみます。

パラメーター

- **level**: **number**
  表示する最大のグループレベル。

継承元 **FlexGrid**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

継承元 **FlexGrid**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

継承元 **FlexGrid**
戻り値 **boolean**

focus

```
focus(): void
```

グリッド全体をスクロールしてビューに入れることなくグリッドにフォーカスを設定するようにオーバーライドされます。

継承元 **FlexGrid**
戻り値 **void**

getBindingColumn

```
getBindingColumn(p: GridPanel, r: number, c: number): Column
```

データ項目をグリッドセルに連結するために使用される**Column** オブジェクトを取得します。

パラメーター

- **p**: **GridPanel**
  セルが含まれる**GridPanel**。
- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

戻り値 **Column**

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セル要素の範囲（ビューポート座標単位）を取得します。

このメソッドは、**cells** パネル内のセル（スクロール可能なデータセル）の範囲を返します。その他のパネルにあるセルの範囲を取得するには、該当する**GridPanel** オブジェクトの**getCellBoundingRect** メソッドを使用してください。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

継承元 **FlexGrid**
戻り値 **Rect**

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

グリッドのスクロール可能領域内のセルに格納された値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

継承元 **FlexGrid**
戻り値 **any**

getClipString

```
getClipString(rng?: CellRange): string
```

CellRange の内容を、クリップボードへのコピーに適した文字列として取得します。

非表示の行および列はクリップボード文字列に含まれません。

パラメーター

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **string**

getColumn

```
getColumn(name: string): Column
```

名前または連結に基づいて列を取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **string**
  検索する名前または連結。

戻り値 **Column**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

GridPanel 内のセルの結合範囲を示す**CellRange** を取得します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。

- **r**: **number**
  セルを含む行のインデックス。

- **c**: **number**
  セルを含む列のインデックス。

- **clip**: **boolean** OPTIONAL
  結合範囲をグリッドの現在のビュー範囲にクリップするかどうか。

継承元 **FlexGrid**
戻り値 **CellRange**

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

パラメーター

- **r**: **number**
  検査するセルの行インデックス。

- **c**: **number**
  検査するセルの列インデックス。

継承元 **FlexGrid**
戻り値 **SelectedState**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

指定されたポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

次に例を示します。

```
// ユーザーがグリッドをクリックしたときに、ポイントをヒットテストします
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

パラメーター

- **pt**: **any**
  調べる**Point** （ページ座標単位）、MouseEventオブジェクト、またはポイントのX座標。

- **y**: **any** OPTIONAL
  ポイントのY座標（ページ座標単位、最初のパラメーターが数値の場合）。

継承元 **FlexGrid**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

指定したCellRangeがこのグリッドの行および列コレクションに対して有効かどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックする範囲。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

autoSizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

autoSizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

autoSizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

autoSizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

beginningEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

cellEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

cellEditEnding イベントを発生させます。

パラメーター

- **e**: **CellEditEndingEventArgs**
  イベントデータを含む**CellEditEndingEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCollapsedHeadersChanged

```
onCollapsedHeadersChanged(e?: EventArgs): void
```

collapsedHeadersChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onCollapsedHeadersChanging

```
onCollapsedHeadersChanging(e: CancelEventArgs): boolean
```

collapsedHeadersChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

戻り値 **boolean**

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

copied イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

copying イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

deletedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

deletingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

draggedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

draggedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

draggingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

draggingColumnOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

draggingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

draggingRowOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

groupCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

groupCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onItemsSourceChanging

```
onItemsSourceChanging(e: CancelEventArgs): boolean
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

loadedRows イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

loadingRows イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

pasted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

pastedCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

pasting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

pastingCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

prepareCellForEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

resizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

resizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

resizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

resizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

rowAdded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

rowEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

rowEditEnding イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

rowEditStarted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

rowEditStarting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

scrollPositionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

selectionChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

sortedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

sortingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

updatedLayout イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

updatingLayout イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

updatingView イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

refresh

```
refresh(fullUpdate?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

継承元 **FlexGrid**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean**
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

- **recycle**: **boolean** OPTIONAL
  既存の要素を再利用するかどうか。

- **state**: **boolean** OPTIONAL
  既存の要素を保持してその状態を更新するかどうか。

継承元 **FlexGrid**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

scrollIntoView

```
scrollIntoView(r: number, c: number, refresh?: boolean): boolean
```

指定したセルが画面に入るようにグリッドをスクロールします。

負の行と列のインデックスは無視されるので、以下を呼び出すと、

```
grid.scrollIntoView(200, -1);
```

グリッドは200行目を表示するように垂直にスクロールしますが、水平にスクロールしません。

パラメーター

- **r**: **number**
  画面に入るようにスクロールする行のインデックス
- **c**: **number**
  画面に入るようにスクロールする列のインデックス。
- **refresh**: **boolean** OPTIONAL
  スクロールした新しい位置をすぐ表示するためにグリッドを更新する必要があるかどうかを決定するオプションのパラメータ。

継承元 **FlexGrid**
戻り値 **boolean**

select

```
select(rng: any, show?: any): void
```

セル範囲を選択し、オプションでそのセル範囲が画面に入るようにスクロールします。

select メソッドは、**CellRange** 、と新しい選択範囲を画面に入るようにスクロールするかどうかを示すオプションのブール型パラメータを渡すことで呼び出すことができます。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(new CellRange(1, 1), true);

// 選択範囲（1,1）~（2,4）を指定し、画面に入るようにスクロールしません。
grid.select(new CellRange(1, 1, 2, 4), false);
```

select メソッドを呼び出してインデックスまたは選択する行と列を渡すこともできます。この場合、選択範囲が画面に入るようにスクロールします。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(1, 1);
```

パラメーター

- **rng**: **any**
  選択する範囲。
- **show**: **any** OPTIONAL
  新しい選択範囲が画面に入るようにスクロールするかどうか。

継承元 **FlexGrid**
戻り値 **void**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

グリッドのスクロール可能領域内のセルの値を設定します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **value**: **any**
  セルに格納する値。
- **coerce**: **boolean** OPTIONAL
  列のデータ型に合わせて値を自動的に変更するかどうか。
- **invalidate**: **boolean** OPTIONAL
  グリッドを無効にして変更を表示するかどうか。

継承元 **FlexGrid**
戻り値 **boolean**

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

文字列を行および列に解析し、その内容を特定の範囲に適用します。

非表示の行および列はスキップされます。

パラメーター

- **text**: **string**
  グリッドに解析する、タブと改行で区切られたテキスト。

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **void**

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

指定されたセルの編集を開始します。

FlexGrid の編集は、Excelの編集によく似ています。 ［F2］キーを押すか、セルをダブルクリックすると、 グリッドが**完全編集** モードになります。 このモードでは、ユーザーが［Enter］、［Tab］、または［Esc］キーを押すか、マウスを使用して選択範囲を移動するまで、セルエディタはアクティブのままになります。完全編集モードでは、カーソルキーを押しても、グリッドの編集モードは終了しません。

テキストをセルに直接入力すると、グリッドは**クイック編集**モードになります。 このモードでは、ユーザーがEnter、Tab、Esc、またはいずれかの矢印キーを押すまで、セルエディタはアクティブなままになります。

通常、完全編集モードは既存の値を変更する際に使用します。 クイック編集モードは新しいデータをすばやく入力する際に使用します。

編集中に［F2］キーを押すことで、完全編集モードとクイック編集モードを切り替えることができます。

パラメーター

- **fullEdit**: **boolean** OPTIONAL
  ユーザーがカーソルキーを押したときも編集モードのままにするかどうか。 デフォルトはtrueです。

- **r**: **number** OPTIONAL
  編集する行のインデックス。 デフォルトは現在選択されている行です。

- **c**: **number** OPTIONAL
  編集する列のインデックス。 デフォルトは現在選択されている列です。

- **focus**: **boolean** OPTIONAL
  編集開始時に、エディタにフォーカスを与えるかどうか。 デフォルトはtrueです。

継承元 **FlexGrid**
戻り値 **boolean**

toggleDropDownList

```
toggleDropDownList(): void
```

現在選択されているセルに関連付けられたドロップダウンリストボックスを切り替えます。

このメソッドでは、セルが編集モードに入ったとき、またはユーザが特定のキーを押したときに、ドロップダウンリストを自動的に表示することができます。

たとえば、このコードでは、グリッドが編集モードに入るたびに、グリッドにドロップダウンリストが表示されます。

```
// グリッドが編集モードに入ると、ドロップダウンリストを表示する。
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

このコードでは、ユーザーがスペースバーを押した場合、グリッドが編集モードに入った後で、ドロップダウンリストが表示されます。

```
// ユーザーがスペースバーを押したときにドロップダウンリストを表示する。
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

継承元 **FlexGrid**
戻り値 **void**

イベント

autoSizedColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizedRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

beginningEdit

セルが編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnded

セル編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnding

セル編集が終了するときに発生します。

このイベントを使用して検証を実行し、無効な編集を防ぐことができます。 たとえば、次のコードは、ユーザーが文字「a」を含まない値を入力できないようにします。このコードは、編集が適用される前に、編集前と編集後の値を取得する方法を示しています。

```
function cellEditEnding (sender, e) {

  // 編集前と編集後の値を取得します
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // newValに「a」が含まれていない場合は、編集をキャンセルします
  e.cancel = newVal.indexOf('a') < 0;
}
```

cancel パラメータをtrueに設定すると、編集された値が無視されて、グリッドでセルの元の値が維持されます。

また、**stayInEditMode** パラメータをtrueに設定すると、グリッドの編集モードが維持され、編集をコミットする前にユーザーが無効な値を修正できます。

継承元 **FlexGrid**
引数 **CellEditEndingEventArgs**

collapsedHeadersChanged

collapsedHeaders プロパティの値が変化した後に発生します。

引数 **EventArgs**

collapsedHeadersChanging

collapsedHeaders プロパティの値が変化した後に発生します。

引数 **CancelEventArgs**

copied

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーした後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

copying

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーするときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラでコピー操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletedRow

ユーザーが［Del］キーを押して行を削除した後に発生します （**allowDelete** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletingRow

ユーザーが［Delete］キーを押して選択されている行を削除するときに発生します（**allowDelete** プロパティを参照）。

イベントハンドラで行の削除をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedColumn

ユーザーが列のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedRow

ユーザーが行のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumn

ユーザーが列のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumnOver

ユーザーが列を別の位置にドラッグするときに発生します。

ハンドラは、このイベントをキャンセルして、ユーザーが特定の位置に列をドロップしないようにすることができます。次に例を示します。

```
// ドラッグされている列を記憶します
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// 'sales'列がインデックス0にドラッグされないようにします
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRow

ユーザーが行のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRowOver

ユーザーが行を別の位置にドラッグするときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

formatItem

セルを表す要素が作成されたときに発生します。

このイベントを使用してセルを表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、グループ行のセルから'wj-wrap'クラスを削除します。

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

継承元 **FlexGrid**
引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

groupCollapsedChanged

グループが展開または折りたたまれた後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

groupCollapsedChanging

グループが展開または折りたたまれる直前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

itemsSourceChanged

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

itemsSourceChanging

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

loadedRows

グリッド行がデータソースの項目に連結された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

loadingRows

グリッド行がデータソースの項目に連結される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

pasted

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastedCell

ユーザーがクリップボードの内容をセルに貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pasting

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた時に発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastingCell

ユーザーがクリップボードの内容をセルに貼り付けるときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

prepareCellForEdit

エディタセルが作成されたとき、それがアクティブになる前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

resizedColumn

ユーザーが列のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizedRow

ユーザーが行のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingColumn

列がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingRow

行がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowAdded

ユーザーが新規行テンプレートを編集して新しい項目を作成したときに発生します（**allowAddNew** プロパティを参照）。

イベントハンドラで新しい項目の内容をカスタマイズしたり、新しい項目の作成をキャンセルしたりできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnded

行編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnding

行編集が終了するとき、変更が確定またはキャンセルされる前に発生します。

このイベントを**rowEditStarted**イベントと組み合わせて使用して、ディープ連結編集を元に戻す操作を実装できます。次に例を示します。

```
// 編集の開始時にディープ連結値を保存します
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.')> -1) { // ディープ連結
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// 編集がキャンセルされたときにディープ連結値を復元します
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // ユーザーによって編集がキャンセルされました
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarted

行が編集モードに入った後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarting

行が編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

scrollPositionChanged

コントロールがスクロールされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

selectionChanged

選択が変更された後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

selectionChanging

選択が変更される前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortedColumn

ユーザーが列ヘッダをクリックしてソートを適用した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortingColumn

ユーザーが列ヘッダをクリックしてソートを適用する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

updatedLayout

グリッドで内部レイアウトが更新された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

updatedView

グリッドが現在のビューを構成する要素の作成/更新を完了したときに発生します。

グリッドのビューは以下のような操作に応じて更新されます。

- グリッドまたはそのデータソースの更新
- 行または列の追加、削除、変更
- グリッドのサイズ変更またはスクロール
- 選択の変更

継承元 **FlexGrid**
引数 **EventArgs**

updatingLayout

グリッドで内部レイアウトが更新される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

updatingView

現在のビューを構成する要素の作成/更新をグリッドが開始したときに発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

wijmo.input モジュール

ファイル `wijmo.input.js`
モジュール **wijmo.input**

文字列、数値、日付、時刻、色を入力するためのコントロールを定義します。

クラス

- AutoComplete
- Calendar
- ColorPicker
- ComboBox
- DropDown
- FormatItemEventArgs
- InputColor
- InputDate
- InputDateTime
- InputMask
- InputNumber
- InputTime
- ListBox
- Menu
- MultiAutoComplete
- MultiSelect
- Popup

列挙体

- DateSelectionMode
- PopupTrigger

AutoComplete クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **ComboBox**
派生クラス **MultiAutoComplete**, **WjAutoComplete**
表示 ☐継承されたメンバー ☐イベント発生元

AutoComplete コントロールは、ユーザー入力時に入力文字列に従って呼び出し元で項目リストをカスタマイズできる入力コントロールです。

このコントロールは**ComboBox** に似ていますが、項目ソースが静的なリストではなく関数（**itemsSourceFunction** ）である点が異なります。たとえば、ユーザーの入力に従ってリモートデータベースの項目を検索できます。

次の例では、**AutoComplete** コントロールを作成し、'countries'配列を使用してリストの内容を設定します。ユーザーが文字を入力すると自動的に国が検索され、候補が絞り込まれて現在の入力と一致するものだけになります。

サンプル

Show me (https://jsfiddle.net/Wijmo5/8HnLx)

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- collectionView
- controlTemplate
- cssMatch
- delay
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- headerPath
- hostElement
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceFunction
- listBox
- maxDropDownHeight
- maxDropDownWidth
- maxItems

- minLength
- placeholder
- rightToLeft
- searchMemberPath
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- text

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus

- refreshed
- refreshing
- selectedIndexChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?: any): AutoComplete
```

AutoComplete クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **AutoComplete**

プロパティ

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

cssMatch

検索語と一致するすべての部分をコンテンツで強調表示するために使用する CSSクラスの名前を取得または設定します。

型 **string**

delay

キーストロークが発生してから検索が実行されるまでの遅延（ミリ秒単位）を取得または設定します。

The default value for this property is **500** milliseconds.

型 **number**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 **ComboBox**
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **DropDown**
型 **HTMLInputElement**

- isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **ComboBox** control, and to true on the **AutoComplete** control.

継承元 **ComboBox**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **`ComboBox`**, **`InputDate`**, **`InputTime`**, **`InputDateTime`**, and **`InputColor`**. It defaults to false for the **`AutoComplete`** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

itemsSourceFunction

ユーザーの入力に従ってリスト項目を動的に提供する関数を取得または設定します。

この関数は以下の3つのパラメーターをとります。

- ユーザーが入力したクエリー文字列
- 返す項目の最大数
- 結果が取得されたときに呼び出すコールバック関数

例:

```
autoComplete.itemsSourceFunction = function (query, max, callback) {

  // サーバーから結果を取得します。
  var params = { query: query, max: max };
  $.getJSON('companycatalog.ashx', params, function (response) {

    // コントロールに結果を返します。
    callback(response);
  });
};
```

型 **Function**

listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

maxItems

ドロップダウンリストに表示する項目の最大数を取得または設定します。

The default value for this property is **6**.

型 **number**

minLength

オートコンプリート候補を検索するために必要な入力の最小長さを取得または設定します。

The default value for this property is **2**.

型 **number**

placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

searchMemberPath

項目の検索時に使用するプロパティのカンマ区切りリストを含む文字列を取得または設定します。

デフォルトでは、**AutoComplete** コントロールは、**displayMemberPath** プロパティで指定された プロパティと比較して一致を検索します。**searchMemberPath** プロパティを使用すると、 追加のプロパティを使用して検索を行うことができます。

たとえば、以下のコードは、会社名を表示し、 会社名、シンボル、および国に基づいて検索を行います。

```
var ac = new wijmo.input.AutoComplete('#autoComplete', {
  itemsSource: companies,
  displayMemberPath: 'name',
  searchMemberPath: 'symbol,country'
});
```

型 **string**

selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValue

selectedValuePath を使用して取得された**selectedItem** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValuePath

selectedValue を**selectedItem** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **ListBox** should include group header items to delimit data groups.

Data groups are created by modifying the **groupDescriptions** property of the **ICollectionView** object used as an **itemsSource**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

Calendar クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **Control**
派生クラス **WjCalendar**
表示 ☐継承されたメンバー ☐イベント発生元

Calendar コントロールには月間カレンダーが表示され、ユーザーはここから 日付を選択することができます。

min プロパティと**max** プロパティを使用すると、 ユーザーが選択できる日付の範囲を制限できます。

min および**max** プロパティの使用方法については、 「minおよびmaxプロパティの使用」トピックを参照してください。

現在選択されている日付を取得または設定するには、**value** プロパティを使用します。

selectionMode プロパティを使用すると、ユーザーが日や月を選択できるか、 またはどの値も選択できないかを決定できます。

Calendar コントロールは、次のキーボードコマンドをサポートしています。

キーの組み合わせ	**アクション**
←	前の日
→	次の日
↑	前の週
↓	次の週
PgUp	前の月
PgDn	次の月
Alt + PgUp	前の年
Alt + PgDn	次の年
Home	月の最初の有効な日
End	月の最後の有効な日
Alt + End	今日

次の例は、**InputDate** コントロールと**InputTime** コントロールを使用して、 日時の情報を含む**Date**値を表示します。どちらのコントロールも 同じコントローラー変数に連結されており、各コントロールがそれぞれの情報（日付または時刻） を編集することに注目してください。この例では、ユーザーが1回のクリックで日付を選択できる **Calendar** コントロールも示されています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/vgc3Y)

コンストラクタ

- constructor

プロパティ

- controlTemplate
- displayMonth
- firstDayOfWeek
- formatDayHeaders
- formatDays
- formatMonths
- formatYear
- formatYearMonth
- hostElement
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemFormatter

- itemValidator
- max
- min
- monthView
- repeatButtons
- rightToLeft
- selectionMode
- showHeader
- showYearPicker
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onDisplayMonthChanged
- onFormatItem
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- displayMonthChanged
- formatItem
- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): Calendar
```

Calendar クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **Calendar**

プロパティ

STATIC controlTemplate

Calendar コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

displayMonth

カレンダーに表示されている月を取得または設定します。

型 **Date**

firstDayOfWeek

週の最初の曜日（カレンダーの最初の列に表示される曜日）を表す値を取得または設定します。

このプロパティをnullに設定すると、現在のカルチャのデフォルトが使用されます。週の最初の曜日は、英語カルチャでは日曜日（0）であり、ほとんどのヨーロッパカルチャでは月曜日（1）です。

型 **number**

formatDayHeaders

月表示で、曜日の上に表示されるヘッダーの書式を取得または設定します。

このプロパティのデフォルト値は**'ddd'**です。

型 **string**

formatDays

月表示で、日の表示書式を取得または設定します。

このプロパティのデフォルト値は 'd 'です（'d'の後のスペースは、短い日付パターンを表す標準な書式'd'と解釈されないようにします）。

型 **string**

formatMonths

年表示で、月の表示書式を取得または設定します。

このプロパティのデフォルト値は**'MMM'**です。

型 **string**

formatYear

年表示で、月の上に表示される年の書式を取得または設定します。

このプロパティのデフォルト値は**'yyyy'**です。

型 **string**

formatYearMonth

月表示で、カレンダーの上に表示される月と年の書式を取得または設定します。

このプロパティのデフォルト値は**'y'**です。

型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **`Control`**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **`Control`**
型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

itemFormatter

カレンダーの日付をカスタマイズするフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意の日付に任意の内容を追加できます。そのため、カレンダーの外観と動作を全面的にカスタマイズすることが可能です。

この関数は以下の2つのパラメーターをとります。

- 書式設定する日付
- 日付を表すHTML要素

以下のサンプルコードは週末を無効な状態で表示します。

```
calendar.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

型 **Function**

itemValidator

日付が選択可能かどうかを決定するバリデーター関数を取得または設定します。

このバリデーター関数は、調べる日付を表す1つのパラメーターを受け取り、その日付が無効で選択できない場合、falseを返す必要があります。

以下のサンプルコードは、週末を無効な状態で表示し、ユーザーがそれらの日付を選択できないようにします。

```
calendar.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

型 **Function**

max

ユーザーがカレンダーで選択できる最も遅い日付を取得または設定します。

The default value for this property is **null**, which means no latest date is defined.

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

型 **Date**

- min

ユーザーがカレンダーで選択できる最も早い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined.

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

型 **Date**

- monthView

カレンダーに1か月と1年のどちらを表示するかを示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- repeatButtons

カレンダーボタンがリピートボタン(押された状態で繰り返し実行するボタン)として動作するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectionMode

ユーザーが日や月を選択できるか、またはどの値も選択できないかを示す値を取得または設定します。

The default value for this property is **DateSelectionMode.Day**.

型 **DateSelectionMode**

- showHeader

現在の月とナビゲーションボタンを含むヘッダ領域をコントロールに表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- showYearPicker

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

value

現在選択されている日付を取得または設定します。

The default value for this property is the current date.

型 **Date**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**`Control`** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**`dispose`** と **`removeEventListener`** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **`Improving scrolling performance with passive listeners`**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **`Control`**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onDisplayMonthChanged

```
onDisplayMonthChanged(e?: EventArgs): void
```

displayMonthChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

displayMonthChanged

displayMonth プロパティが変更された後に発生します。

引数 **EventArgs**

formatItem

カレンダーの日を表す要素が作成されたときに発生します。

このイベントを使用してカレンダーの項目を表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、**formatItem** イベントを使用して週末を無効な状態にし、 カレンダーにグレーで表示されるようにします。

```
// 日曜日と土曜日を無効にします。
calendar.formatItem.addHandler(function (s, e) {
  var day = e.data.getDay();
  if (day == 0 || day == 6) {
    wijmo.addClass(e.item, 'wj-state-disabled');
  }
});
```

引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

引数 **EventArgs**

ColorPicker クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **Control**
派生クラス **WjColorPicker**
表示 ☐ 継承されたメンバー ☐ イベント発生元

ColorPicker コントロールには色チャネル（色相、彩度、明度、アルファ）を調整するパネルが表示され、ユーザーはこのパネルをクリックして色を選択できます。現在選択されている色を取得または設定するには、**value** プロパティを使用します。このコントロールは、**InputColor** コントロールのドロップダウンとして使用されます。

サンプル

Show me (https://jsfiddle.net/Wijmo5/z84ebpec)

コンストラクタ

- constructor

プロパティ

- controlTemplate
- hostElement
- isDisabled
- isTouching
- isUpdating
- palette
- rightToLeft
- showAlphaChannel
- showColorString
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): ColorPicker
```

ColorPicker クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **ColorPicker**

プロパティ

STATIC controlTemplate

ColorPicker コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- palette

パレットの色を含む配列を取得または設定します。このパレットは10色で構成され、10個の文字列の配列によって表されます。最初の2色は通常、白と黒です。

型 **string**[]

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- showAlphaChannel

ユーザーが**`ColorPicker`** を使用して色のアルファチャネル（透明度）を編集できるかどうかを示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- showColorString

`ColorPicker` に現在の色の文字列表現を表示するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- value

現在選択されている色を取得または設定します。 このプロパティのデフォルトは「white」です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

引数 **EventArgs**

ComboBox クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **DropDown**
派生クラス **AutoComplete**, **InputTime**, **Menu**, **MultiSelect**, **WjComboBox**
表示 ☐継承されたメンバー ☐イベント発生元

ComboBox コントロールでは、リストから文字列を選択できます。

このコントロールは、ユーザーのキー入力に伴って自動的にエントリを補完し、利用可能な項目を ドロップダウンリストに表示することができます。

itemsSource プロパティを使用して、オプションのリストに項目を挿入します。 項目には、文字列またはオブジェクトを使用できます。項目がオブジェクトの場合は、 **displayMemberPath** を使用してリストに表示する項目のプロパティを定義し、 **selectedValuePath** プロパティを使用して コンボボックスの**selectedValue** プロパティの設定に使用する 項目のプロパティを定義します。

selectedIndex プロパティまたは**text** プロパティを使用して、 現在されている項目を特定します。

isEditable プロパティは、リストにない値をユーザーが入力できるかどうかを 決定します。

以下の例では、**ComboBox** コントロールを作成し、国のリストを項目として挿入します。ユーザーが文字を入力すると、自動的に国が検索されます。 **isEditable** プロパティはfalseに設定されているため、ユーザーは リスト内のいずれかの項目を選択しなければなりません。

この例ではさらに、HTML の **<select>** 要素を使用して**ComboBox** を作成し、**<option>** 子要素でリストの内容を設定する方法も示します。

サンプル

Show me (https://jsfiddle.net/Wijmo5/8HnLx)

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- headerPath
- hostElement
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox

- maxDropDownHeight
- maxDropDownWidth
- placeholder
- rightToLeft
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- text

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus

- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?): ComboBox
```

ComboBox クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **ComboBox**

プロパティ

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

型 **Event**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **DropDown**
型 **HTMLInputElement**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **ComboBox** control, and to true on the **AutoComplete** control.

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

型 **Function**

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

型 **any**

listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

型 **ListBox**

maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

型 **number**

- maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

型 **number**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

型 **number**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

型 **any**

- selectedValue

selectedValuePath を使用して取得された**selectedItem** の値を取得または設定します。

型 **any**

- selectedValuePath

selectedValue を**selectedItem** から取得するために使用するプロパティの名前を 取得または設定します。

型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

showGroups

Gets or sets a value that determines whether the drop-down **ListBox** should include group header items to delimit data groups.

Data groups are created by modifying the **groupDescriptions** property of the **ICollectionView** object used as an **itemsSource**.

The default value for this property is **false**.

型 **boolean**

text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

DropDown クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **Control**
派生クラス **ComboBox**, **InputColor**, **InputDate**
表示 ☐継承されたメンバー ☐イベント発生元

DropDownコントロール（抽象）。

入力要素と、ドロップダウンを表示/非表示にするボタンで構成されます。

派生クラスでは、_createDropDownメソッドをオーバーライドして、 ドロップダウン領域に表示するエディタ（項目のリスト、カレンダー、色エディタなど）を作成する必要があります。

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- controlTemplate
- dropDown
- dropDownCssClass
- hostElement
- inputElement
- isAnimated
- isDisabled
- isDroppedDown
- isReadOnly
- isRequired
- isTouching
- isUpdating
- placeholder
- rightToLeft
- showDropDownButton
- text

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll

- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onRefreshed
- onRefreshing
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- refreshed
- refreshing
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?): DropDown
```

DropDown クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **DropDown**

プロパティ

- autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

dropDown

`isDroppedDown` プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **`Control`**
型 **HTMLElement**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

型 **HTMLInputElement**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **`Control`**
型 **boolean**

isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **`ComboBox`**, **`InputDate`**, **`InputTime`**, **`InputDateTime`**, and **`InputColor`**. It defaults to false for the **`AutoComplete`** control.

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

text

コントロールに表示されるテキストを取得または設定します。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

引数 **EventArgs**

FormatItemEventArgs クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

formatItem イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- data
- empty
- index
- item

コンストラクタ

constructor

```
constructor(index: number, data: any, item: HTMLElement): FormatItemEventArgs
```

FormatItemEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **index**: **number**
  Index of the item being formatted in the source **ICollectionView**, or -1 if the item is a group header.
- **data**: **any**
  Data item being formatted, or a **CollectionViewGroup** object if the item is a group header.
- **item**: **HTMLElement**
  書式設定されるリスト項目を表す要素。

戻り値 **FormatItemEventArgs**

プロパティ

- data

書式設定されるデータ項目を取得します。

型 **any**

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

- index

書式設定される項目のインデックスを取得します。

型 **number**

- item

書式設定されるリスト項目を表す要素への参照を取得します。

型 **HTMLElement**

InputColor クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **DropDown**
派生クラス **WjInputColor**
表示 ☐継承されたメンバー ☐イベント発生元

InputColor コントロールを使用すると、ユーザーはHTMLで使用可能な色文字列を入力して色を選択するか、**ColorPicker** コントロールを表示するドロップダウンから色を選択できます。現在選択されている色を取得または設定するには、**value** プロパティを使用します。

サンプル

Show me (https://jsfiddle.net/Wijmo5/z84ebpec)

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- colorPicker
- controlTemplate
- dropDown
- dropDownCssClass
- hostElement
- inputElement
- isAnimated
- isDisabled
- isDroppedDown
- isReadOnly
- isRequired
- isTouching
- isUpdating
- palette
- placeholder
- rightToLeft
- showAlphaChannel
- showDropDownButton
- text
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus

- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onRefreshed
- onRefreshing
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- refreshed
- refreshing
- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputColor
```

InputColor クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **InputColor**

プロパティ

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

colorPicker

ドロップダウンに表示される**`ColorPicker`** コントロールへの参照を取得します。

型 **`ColorPicker`**

STATIC controlTemplate

`DropDown` コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

dropDown

`isDroppedDown` プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **`Control`**
型 **HTMLElement**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **DropDown**
型 **HTMLInputElement**

- isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **`ComboBox`**, **`InputDate`**, **`InputTime`**, **`InputDateTime`**, and **`InputColor`**. It defaults to false for the **`AutoComplete`** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- palette

パレットの色を含む配列を取得または設定します。このパレットは10色で構成され、10個の文字列の配列によって表されます。最初の2色は通常、白と黒です。

型 **string**[]

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 `DropDown`
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- showAlphaChannel

ユーザーが`ColorPicker` を使用して色のアルファチャネル（透明度）を編集できるかどうかを示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 `DropDown`
型 **boolean**

- text

コントロールに表示されるテキストを取得または設定します。

型 **string**

- value

現在の色を取得または設定します。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

引数 **EventArgs**

InputDate クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **DropDown**
派生クラス **InputDateTime**, **WjInputDate**
表示 ☐継承されたメンバー ☐イベント発生元

InputDate コントロールでは、ユーザーが**Globalize** クラスでサポートされている書式を使用して日付を入力するか、 **Calendar** コントロールに含まれるドロップダウンから日付を選択できます。

ユーザーが入力できる日付の範囲を制限するには、**min** プロパティと**max** プロパティを使用します。

min および**max** プロパティの使用方法については、 「minおよびmaxプロパティの使用」トピックを参照してください。

現在選択されている日付を取得または設定するには、**value** プロパティを使用します。

次の例では、**InputDate** コントロールと**InputTime** コントロールを使用して**Date**値（日付と時刻の情報を含む）を表示します。どちらのコントロールも 同じコントローラー変数に連結されており、各コントロールがそれぞれの情報（日付または時刻） を編集することに注目してください。この例では、 1 回のクリックで日付を選択できる**Calendar** コントロールも示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/vgc3Y)

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- calendar
- controlTemplate
- dropDown
- dropDownCssClass
- format
- hostElement
- inputElement
- inputType
- isAnimated
- isDisabled
- isDroppedDown
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- mask
- max
- min
- placeholder
- repeatButtons
- rightToLeft
- selectionMode

- showDropDownButton
- showYearPicker
- text
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onRefreshed
- onRefreshing
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- refreshed
- refreshing
- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputDate
```

InputDate クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **InputDate**

プロパティ

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

calendar

ドロップダウンボックスに表示される **Calendar** コントロールへの参照を取得します。

型 **Calendar**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

format

選択された日付の表示に使用される書式を取得または設定します。

書式文字列は、.NET形式の **日付書式文字列 (http://msdn.microsoft.com/ja-JP/library/8kb3ddd4(v=vs.110).aspx)**として表されます。 The default value for this property is **d**, the culture-dependent short date pattern (e.g. 6/15/2020 in the US, 15/6/2020 in France, or 2020/6/15 in Japan).

型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

型 **HTMLInputElement**

inputType

コントロールによってホストされているHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは"tel"に設定されており、マイナス記号と小数点記号を含む数値キーパッドが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"または"text"に変更してみてください。

"number"タイプのような入力要素はChromeで選択を防ぐため、推奨されません。詳細については、http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome を参照してください。

型 **string**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **`ComboBox`**, **`InputDate`**, **`InputTime`**, **`InputDateTime`**, and **`InputColor`**. It defaults to false for the **`AutoComplete`** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

ドロップダウンカレンダーの日付をカスタマイズするフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意の日付に任意の内容を追加できます。そのため、カレンダーの外観と動作を全面的にカスタマイズすることが可能です。

この関数は以下の2つのパラメーターをとります。

- 書式設定する日付
- 日付を表すHTML要素

以下のサンプルコードは週末を無効な状態で表示します。

```
inputDate.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

型 **Function**

itemValidator

日付が選択可能かどうかを決定するバリデーター関数を取得または設定します。

このバリデーター関数は、調べる日付を表す1つのパラメーターを受け取り、その日付が無効で選択できない場合、falseを返す必要があります。

以下のサンプルコードは、週末の日付を選択できないようにします。

```
inputDate.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

型 **Function**

mask

編集中に使用するマスクを取得または設定します。 マスクの書式は、**InputMask** コントロールで使用される書式と同じです。 指定する場合、このマスクは**format** プロパティの値と互換性がある必要があります。 たとえば、'MM/dd/yyyy'と書式設定された 日付の入力にマスク'99/99/9999'を使用できます。

型 **string**

max

ユーザーが入力できる最も遅い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined. **min** および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

型 **Date**

- min

ユーザーが入力できる最も早い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined. **min** および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

型 **Date**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- repeatButtons

カレンダーボタンがリピートボタン(押された状態で繰り返し実行するボタン)として動作するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectionMode

ユーザーが日や月を選択できるか、またはどの値も選択できないかを示す値を取得または設定します。

このプロパティは、ドロップダウンカレンダーの動作に影響しますが、 日付の表示に使用する書式には影響しません。 **selectionMode** を'Month'に設定した場合は、通常、 **format** プロパティを'MMM yyyy'などの日を含まない書式に 設定する必要があります。次に例を示します。

```
var inputDate = new wijmo.input.InputDate('#el, {
  selectionMode: 'Month',
  format: 'MMM yyyy'
});
```

The default value for this property is **DateSelectionMode.Day**.

型 **DateSelectionMode**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

showYearPicker

Gets or sets a value that determines whether the drop-down calendar should display a list of years when the user clicks the header element on the year calendar.

The default value for this property is **true**.

型 **boolean**

text

コントロールに表示されるテキストを取得または設定します。

型 **string**

value

現在の日付を取得または設定します。

The default value for this property is the current date.

型 **Date**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

引数 **EventArgs**

InputDateTime クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **InputDate**
派生クラス **WjInputDateTime**
表示 ☐継承されたメンバー ☐イベント発生元

InputDateTime コントロールは、**InputDate** コントロールを拡張して、 **Globalize** クラスでサポートされている任意の書式で完全な日付/時刻の値を キー入力するか、あるいはドロップダウンカレンダーから日付を、ドロップダウンリストから 時刻を選択することで、日付と時刻を入力できるようにします。

min プロパティと**max** プロパティを使用すると、 ユーザーが入力できる日付の範囲を制限できます。

timeMin プロパティと**timeMax** プロパティを使用すると、 ユーザーが入力できる時刻の範囲を制限できます。

value プロパティを使用すると、現在選択されている日時を取得または設定できます。

次の例は、**InputDateTime** コントロールにより、単一のコントロールを使用して日付と時刻を編集する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/465gmuL2)

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- calendar
- controlTemplate
- dropDown
- dropDownCssClass
- format
- hostElement
- inputElement
- inputTime
- inputType
- isAnimated
- isDisabled
- isDroppedDown
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- mask
- max
- min
- placeholder
- repeatButtons
- rightToLeft
- selectionMode

- showDropDownButton
- showYearPicker
- text
- timeFormat
- timeMax
- timeMin
- timeStep
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onRefreshed
- onRefreshing
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- refreshed
- refreshing
- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputDateTime
```

InputDateTime クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **InputDateTime**

プロパティ

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

calendar

ドロップダウンボックスに表示される **Calendar** コントロールへの参照を取得します。

継承元 **InputDate**
型 **Calendar**

STATIC controlTemplate

InputDateTime コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

- format

選択された日付の表示に使用される書式を取得または設定します。

書式文字列は、.NET形式の **日付書式文字列 (http://msdn.microsoft.com/ja-JP/library/8kb3ddd4(v=vs.110).aspx)**として表されます。 The default value for this property is **d**, the culture-dependent short date pattern (e.g. 6/15/2020 in the US, 15/6/2020 in France, or 2020/6/15 in Japan).

継承元 **InputDate**
型 **string**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **InputDate**
型 **HTMLInputElement**

- inputTime

内部の**InputTime** コントロールへの参照を取得します。これにより、 コントロールの完全なオブジェクトモデルにアクセスできます。

型 **InputTime**

- inputType

コントロールによってホストされているHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは"tel"に設定されており、マイナス記号と小数点記号を含む数値キーパッドが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"または"text"に変更してみてください。

"number"タイプのような入力要素はChromeで選択を防ぐため、推奨されません。詳細については、
http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome を参照してください。

継承元 **InputDate**
型 **string**

- isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **`Control`**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **`DropDown`**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **`DropDown`**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **`ComboBox`**, **`InputDate`**, **`InputTime`**, **`InputDateTime`**, and **`InputColor`**. It defaults to false for the **`AutoComplete`** control.

継承元 **`DropDown`**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

itemFormatter

ドロップダウンカレンダーの日付をカスタマイズするフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意の日付に任意の内容を追加できます。そのため、カレンダーの外観と動作を全面的にカスタマイズすることが可能です。

この関数は以下の2つのパラメーターをとります。

- 書式設定する日付
- 日付を表すHTML要素

以下のサンプルコードは週末を無効な状態で表示します。

```
inputDate.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

継承元 **InputDate**
型 **Function**

itemValidator

日付が選択可能かどうかを決定するバリデーター関数を取得または設定します。

このバリデーター関数は、調べる日付を表す1つのパラメーターを受け取り、その日付が無効で選択できない場合、falseを返す必要があります。

以下のサンプルコードは、週末の日付を選択できないようにします。

```
inputDate.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

継承元 **InputDate**
型 **Function**

mask

編集中に使用するマスクを取得または設定します。 マスクの書式は、**InputMask** コントロールで使用される書式と同じです。 指定する場合、このマスクは**format** プロパティの値と互換性がある必要があります。 たとえば、'MM/dd/yyyy'と書式設定された 日付の入力にマスク'99/99/9999'を使用できます。

継承元 **InputDate**
型 **string**

max

ユーザーが入力できる最も遅い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined. **min** および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputDate**
型 **Date**

- min

ユーザーが入力できる最も早い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined. **min** および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputDate**
型 **Date**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- repeatButtons

カレンダーボタンがリピートボタン(押された状態で繰り返し実行するボタン)として動作するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **InputDate**
型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectionMode

ユーザーが日や月を選択できるか、またはどの値も選択できないかを示す値を取得または設定します。

このプロパティは、ドロップダウンカレンダーの動作に影響しますが、 日付の表示に使用する書式には影響しません。 **selectionMode** を'Month'に設定した場合は、通常、 **format** プロパティを'MMM yyyy'などの日を含まない書式に 設定する必要があります。次に例を示します。

```
var inputDate = new wijmo.input.InputDate('#el, {
  selectionMode: 'Month',
  format: 'MMM yyyy'
});
```

The default value for this property is **DateSelectionMode.Day**.

継承元 **InputDate**
型 **DateSelectionMode**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showYearPicker

Gets or sets a value that determines whether the drop-down calendar should display a list of years when the user clicks the header element on the year calendar.

The default value for this property is **true**.

継承元 **InputDate**
型 **boolean**

- text

コントロールに表示されるテキストを取得または設定します。

継承元 **InputDate**
型 **string**

- timeFormat

ドロップダウンリスト内の時刻の表示に使用される書式を取得または設定します。

このプロパティは、コントロールの入力要素に表示される値には影響しません。 入力要素の値は、**format** プロパティを使用して書式設定されます。

書式文字列は、.NET形式の**時間書式文字列**として表されます。

型 **string**

- timeMax

ユーザーが入力できる最遅時間を取得または設定します。

型 **Date**

- timeMin

ユーザーが入力できる最早時間を取得または設定します。

型 **Date**

- timeStep

時刻のドロップダウンリストのエントリ間の分数を取得または設定します。

型 **number**

- value

現在の日付を取得または設定します。

The default value for this property is the current date.

継承元 **InputDate**
型 **Date**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **InputDate**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

継承元 **InputDate**
引数 **EventArgs**

InputMask クラス

ファイル	`wijmo.input.js`
モジュール	**wijmo.input**
基本クラス	**Control**
派生クラス	**WjInputMask**
表示	☐継承されたメンバー ☐イベント発生元

InputMask コントロールは、ユーザーが何を入力できるかを制御する手段を提供します。

ユーザーが無効なデータを誤って入力することを防いだり、キー入力するときに リテラル（日付のスラッシュなど）をスキップして時間を節約することができます。

入力の検証に使用するマスクは、**mask** プロパティで定義されます。 マスクには、次の特殊文字を使用できます。

0	数字。
9	数字またはスペース。
#	数字、符号、またはスペース。
L	英字。
l	英字またはスペース。
A	英数字。
a	英数字またはスペース。
.	ローカライズされた小数点。
,	ローカライズされた桁区切り。
:	ローカライズされた時刻区切り。
/	ローカライズされた日付の区切り文字。
$	ローカライズされた通貨記号。
<	後続の文字を小文字に変換します。
>	後続の文字を大文字に変換します。
|	大文字小文字の変換を無効にします。
****	任意の文字をエスケープして、リテラルに変えます。
9	DBCS数字。
J	DBCSひらがな。
G	DBCS大文字ひらがな。
K	DBCSカタカナ。
N	DBCS大文字カタカナ。
K	SBCSカタカナ。
N	SBCS大文字カタカナ。
Z	任意のDBCS文字。
H	任意のSBCS文字。
他のすべて	リテラル。

次の例は、**InputMask** コントロールを使用してカスタム形式を含む文字列を編集する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/j6er01bx)

コンストラクタ

- constructor

プロパティ

- controlTemplate
- hostElement
- inputElement
- isDisabled
- isReadOnly
- isRequired
- isTouching
- isUpdating
- mask

- maskFull
- placeholder
- promptChar
- rawValue
- rightToLeft
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputMask
```

InputMask クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **InputMask**

プロパティ

STATIC controlTemplate

InputMask コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

型 **HTMLInputElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- isRequired

コントロール値が空以外の文字列でなければならないかどうかを 示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- mask

ユーザーのキー入力に伴って入力を検証するために使用されるマスクを取得または設定します。このマスクは、**`InputMask`** トピックにリストされているマスク文字を使用した 文字列として定義されます。

型 **string**

- maskFull

マスクに対して完全に文字が入力されたかどうかを示す値を取得します。

型 **boolean**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

型 **string**

- promptChar

コントロール内の入力位置を示すために使用される記号を取得または設定します。

型 **string**

- rawValue

コントロールの未加工の値を取得または設定します（マスクリテラルを除く）。コントロールの未加工の値には、プロンプトとリテラル文字は含まれません。たとえば、**`mask`** プロパティが"AA-9999"に設定されている場合に、ユーザーが値"AB-1234"を入力すると、**`rawValue`** プロパティはマスクに含まれるハイフンを除外して"AB1234"を返します。

型 **string**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

value

コントロールに現在表示されているテキストを取得または設定します。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この`Control` が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（`dispose` と `removeEventListener` メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see `Improving scrolling performance with passive listeners`.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 `Control`
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

引数 **EventArgs**

InputNumber クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **Control**
派生クラス **WjInputNumber**
表示 ☐継承されたメンバー ☐イベント発生元

InputNumber コントロールでは、数値を入力できます。

このコントロールはユーザーが誤って無効なデータを入力することを防ぎ、編集時に数値を書式設定します。

［-］キーを押すと、カーソルの位置に関係なく、編集中の値の記号が反転します。

min プロパティと**max** プロパティを使用すると、入力可能な値の範囲を制限できます。また、**step** プロパティを使用すると、クリックで値を増減できるスピナーボタンを提供できます。

min および**max** プロパティの使用方法については、 「minおよびmaxプロパティの使用」トピックを参照してください。

現在選択されている数値を取得または設定するには、**value** プロパティを使用します。

次の例では、いくつかの**InputNumber** コントロールを作成し、それぞれ異なる書式、範囲、ステップ値を使用した場合の効果を示します。

サンプル

Show me (https://jsfiddle.net/Wijmo5/Cf9L9)

コンストラクタ

- constructor

プロパティ

- controlTemplate
- format
- hostElement
- inputElement
- inputType
- isDisabled
- isReadOnly
- isRequired
- isTouching
- isUpdating
- max
- min
- placeholder
- repeatButtons
- rightToLeft
- showSpinner
- step
- text
- value

メソッド

- addEventListener
- applyTemplate

- beginUpdate
- clamp
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputNumber
```

InputNumber クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **InputNumber**

プロパティ

STATIC controlTemplate

InputNumber コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

format

編集中の数値の表示に使用される書式を取得または設定します（**Globalize** を参照）。

この書式文字列は、.NETスタイルの**標準の数値書式文字列**として表されます。

型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

型 **HTMLInputElement**

inputType

コントロールによってホストされているHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは"tel"に設定されており、マイナス記号と小数点記号を含む数値キーパッドが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"または"text"に変更してみてください。

"number"タイプのような入力要素はChromeで選択を防ぐため、推奨されません。詳細については、http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome を参照してください。

型 **string**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- isRequired

コントロールの値を必ず数値に設定しなければならないか、（コントロールの内容を削除することによって）値をnullに設定できるかを示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- max

ユーザーが入力できる最も大きい数値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

型 **number**

- min

ユーザーが入力できる最も小さい数値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

型 **number**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

型 **string**

- repeatButtons

スピナーボタンがリピートボタン(押された状態で繰り返し実行するボタン)として動作するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- showSpinner

コントロールに値を増減するスピナーボタンを表示するかどうかを示す値を取得または設定します（ステッププロパティを 0 以外の値に設定する必要があります）。

The default value for this property is **true**.

型 **boolean**

- step

ユーザーがスピナーボタンを押したときに**`value`** プロパティを増減する量を取得または設定します。

型 **number**

- text

コントロールに表示されているテキストを取得または設定します。

型 **string**

- value

コントロールの現在の値を取得または設定します。

型 **number**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

clamp

```
clamp(value: number): number
```

min プロパティおよび**max** プロパティで定義された範囲内の値を返します。

パラメーター

- **value**: **number**
  クランプする値。

戻り値 **number**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能性があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

引数 **EventArgs**

InputTime クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **ComboBox**
派生クラス **WjInputTime**
表示 ☐継承されたメンバー ☐イベント発生元

InputTime コントロールを使用すると、**Globalize** クラスでサポートされる任意の書式で時間を入力したり、ドロップダウンリストから時間を選択することができます。

リストに表示される値は、**min** 、**max** 、および**step** プロパティによって決定されます。

min および**max** プロパティの使用方法については、 「minおよびmaxプロパティの使用」トピックを参照してください。

value プロパティは、ユーザーによって選択された時刻を表す**Date** オブジェクトを取得または設定します。

次の例は、**InputDate** コントロールと**InputTime** コントロールを使用して、（日時の情報を含む）**Date**値を表示します。どちらのコントロールも 同じコントローラー変数に連結されており、各コントロールがそれぞれの情報（日付または時刻） を編集することに注目してください。この例では、１回のクリックで日付を選択できる**Calendar** コントロールも示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/vgc3Y)

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- format
- formatItem
- headerPath
- hostElement
- inputElement
- inputType
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- mask

- max
- maxDropDownHeight
- maxDropDownWidth
- min
- placeholder
- rightToLeft
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- step
- text
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputTime
```

InputTime クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **InputTime**

プロパティ

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

format

選択された時刻の表示に使用される書式を取得または設定します（**Globalize** を参照）。

書式文字列は、.NET形式の**時間書式文字列**として表されます。

型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 **ComboBox**
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

型 **HTMLInputElement**

inputType

コントロールによってホストされているHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは"tel"に設定されており、マイナス記号と小数点記号を含む数値キーパッドが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"または"text"に変更してみてください。

"number"タイプのような入力要素はChromeで選択を防ぐため、推奨されません。詳細については、http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome を参照してください。

型 **string**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **ComboBox** control, and to true on the **AutoComplete** control.

継承元 **ComboBox**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

mask

編集時に使用するマスクを取得または設定します。

マスクの書式は**InputMask** コントロールで使用される書式と同じです。

指定する場合、このマスクは**format** プロパティの値と互換性がある必要があります。たとえば、短い時刻（書式't'）の入力に マスク'99:99 >LL'を使用できます。

型 **string**

max

ユーザーが入力できる最遅時間を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

型 **Date**

maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

min

ユーザーが入力できる最早時間を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

型 **Date**

placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValue

selectedValuePath を使用して取得された**selectedItem** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValuePath

selectedValue を**selectedItem** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **ListBox** should include group header items to delimit data groups.

Data groups are created by modifying the **groupDescriptions** property of the **ICollectionView** object used as an **itemsSource**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- step

ドロップダウンリストの項目の間隔（分数）を取得または設定します。

このプロパティのデフォルト値は15分です。 null、ゼロ、または負の値に設定すると、ドロップダウンが無効になります。

型 **number**

- text

コントロールに表示されているテキストを取得または設定します。

型 **string**

- value

現在の入力時間を取得または設定します。

型 **Date**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

引数 **EventArgs**

ListBox クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **Control**
派生クラス **WjListBox**
表示 ☐継承されたメンバー ☐イベント発生元

ListBox コントロールは、プレーンテキストまたはHTMLを含む項目のリストを 表示し、ユーザーがキーボードまたはマウスで項目を選択できるようにします。

selectedIndex プロパティを使用して、現在選択されている項目を 特定します。

ListBox に項目を挿入するには、文字列の配列またはオブジェクトの配列を 使用できます。オブジェクトの配列の場合は、**displayMemberPath** プロパティで、リストに表示するオブジェクトのプロパティを指定します。

プレーンテキストではなくHTMLを保持する項目を表示するには、 **isContentHtml** プロパティをtrueに設定します。

ListBox コントロールは以下のキーボードコマンドをサポートします。

キーの組み合わせアクション
↑/↓ 前の項目/次の項目を選択します。
PageUp/Down 選択項目の1ページ上または1ページ下の項目を選択します。
Home/End 最初の項目/最後の項目を選択します。
Space 現在の項目のチェックボックスを切り替えます(**checkedMemberPath** プロパティを参照)。
その他の文字 入力されたテキスト(複数文字の自動検索)を含む項目を検索します。

次の例は、**ListBox** コントロールを作成した後、 'countries'配列を使用して設定します。ユーザーが選択範囲を移動すると、コントロールはその**selectedIndex** プロパティ と**selectedItem** プロパティを更新します。

サンプル

Show me (https://jsfiddle.net/Wijmo5/8HnLx)

コンストラクタ

- constructor

プロパティ

- checkedItems
- checkedMemberPath
- collectionView
- displayMemberPath
- hostElement
- isContentHtml
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemRole
- itemsSource
- maxHeight
- rightToLeft
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showGroups

showGroups

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getDisplayValue
- getItemChecked
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- isItemEnabled
- loadList
- onCheckedItemsChanged
- onFormatItem
- onGotFocus
- onItemChecked
- onItemsChanged
- onLoadedItems
- onLoadingItems
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- refresh
- refreshAll
- removeEventListener
- setItemChecked
- showSelection
- toggleItemChecked

イベント

- checkedItemsChanged
- formatItem
- gotFocus
- itemChecked
- itemsChanged
- loadedItems
- loadingItems

- lostFocus
- refreshed
- refreshing
- selectedIndexChanged

コンストラクタ

constructor

```
constructor(element: any, options?): ListBox
```

ListBox クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **ListBox**

プロパティ

checkedItems

現在チェックされている項目を含む配列を取得または設定します。

型 **any**[]

checkedMemberPath

各項目の横に配置されるCheckBoxの制御に使用されるプロパティの名前を 取得または設定します。このプロパティを使用して、複数選択ListBoxを作成します。項目をオンまたはオフにすると、**itemChecked** イベントが発生します。オン/オフにした項目を取得するには、**selectedItem** プロパティを使用します。現在オンの項目のリストを取得するには、**checkedItems** プロパティを使用します。

型

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

型 **ICollectionView**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isContentHtml

項目の内容がプレーンテキストとHTMLのどちらであるかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
listBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

型 **Function**

- itemRole

リスト項目に追加されている「role」属性の値を取得または設定します。このプロパティのデフォルト値は 「option」です。

型 **string**

- itemsSource

リスト項目を含む配列または**ICollectionView** オブジェクトを取得または設定します。

型 **any**

- maxHeight

リストの最大の高さを取得または設定します。

型 **number**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectedIndex

現在選択されている項目のインデックスを取得または設定します。

型 **number**

- selectedItem

現在選択されている項目を取得または設定します。

型 **any**

- selectedValue

selectedValuePath を使用して取得された**selectedItem** の値を取得または設定します。

型 **any**

- selectedValuePath

selectedValue を**selectedItem** から取得するために使用するプロパティの名前を 取得または設定します。

型 **string**

showGroups

Gets or sets a value that determines whether the **`ListBox`** should include group header items to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as an **`itemsSource`**.

The **`ListBox`** only shows the first level of grouping.

The default value for this property is **false**.

型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**`Control`** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**`dispose`** と **`removeEventListener`** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **`Improving scrolling performance with passive listeners`**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **`Control`**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index: number): string
```

指定したインデックスにある項目に対して表示されるテキストを（プレーンテキストとして）取得します。

パラメーター

- **index**: **number**
  itemsSource 内の項目のインデックス。

戻り値 **string**

getDisplayValue

```
getDisplayValue(index: number): string
```

指定したインデックスにある項目に対して表示される文字列を取得します。この文字列は、**isContentHtml** プロパティの設定に応じてプレーンテキストまたはHTMLのどちらかになります。

パラメーター

- **index**: **number**
  itemsSource 内の項目のインデックス。

戻り値 **string**

getItemChecked

```
getItemChecked(index: number): boolean
```

リストの項目のオン/オフ状態を取得します。このメソッドは、複数選択ListBoxにのみ適用できます （**checkedMemberPath** プロパティを参照）。

パラメーター

- **index**: **number**
  項目インデックス。

戻り値 **boolean**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(e: HTMLElement): number
```

Gets the data index of an element within the list.

パラメーター

- **e**: **HTMLElement**
  検索する要素。

戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isItemEnabled

```
isItemEnabled(index: number): void
```

指定したインデックスにある項目が有効かどうかを決定する値を取得します。

パラメーター

- **index**: **number**
  itemsSource 内の項目のインデックス。

戻り値 **void**

loadList

```
loadList(): void
```

現在の**itemsSource** からの項目を含むリストをロードします。

戻り値 **void**

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

checkedItemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onItemChecked

```
onItemChecked(e?: EventArgs): void
```

itemChecked イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onItemsChanged

```
onItemsChanged(e?: EventArgs): void
```

itemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLoadedItems

```
onLoadedItems(e?: EventArgs): void
```

loadedItems イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLoadingItems

```
onLoadingItems(e?: EventArgs): void
```

loadingItems イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

setItemChecked

```
setItemChecked(index: number, checked: boolean): void
```

リストの項目のオン/オフ状態を設定します。このメソッドは、複数選択ListBoxにのみ適用できます （**checkedMemberPath** プロパティを参照）。

パラメーター

- **index**: **number**
  項目インデックス。
- **checked**: **boolean**
  項目の新しいチェック状態。

戻り値 **void**

showSelection

```
showSelection(setFocus?: boolean): void
```

選択された項目を強調表示し、画面に入るようにスクロールします。

パラメーター

- **setFocus**: **boolean** OPTIONAL
  Whether to set the focus to the list after scrolling the selected item into view.

戻り値 **void**

toggleItemChecked

```
toggleItemChecked(index: number): void
```

リストの項目のオン/オフ状態を切り替えます。このメソッドは、複数選択ListBoxにのみ適用できます （**checkedMemberPath** プロパティを参照）。

パラメーター

- **index**: **number**
  項目インデックス。

戻り値 **void**

イベント

checkedItemsChanged

checkedItems プロパティの値が変更されたときに発生します。

引数 **EventArgs**

formatItem

リスト項目を表す要素が作成されたときに発生します。このイベントを使用してリストの項目を表示用に書式設定できます。このイベントは、目的は**itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

itemChecked

ユーザーが現在の項目をオンまたはオフにすると発生します。このイベントは、 **checkedMemberPath** にプロパティの名前を設定して、コントロール内の各項目に チェックボックスを追加した場合に発生します。 オン/オフにした項目を取得するには、**selectedItem** プロパティを使用します。

引数 **EventArgs**

itemsChanged

項目のリストが変更されたときに発生します。

引数 **EventArgs**

loadedItems

リスト項目が生成される後に発生します。

引数 **EventArgs**

loadingItems

リスト項目が生成される前に発生します。

引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

引数 **EventArgs**

Menu クラス

ファイル	wijmo.input.js
モジュール	**wijmo.input**
基本クラス	**ComboBox**
派生クラス	**WjMenu**
表示	☐ 継承されたメンバー ☐ イベント発生元

Menu コントロールは、ドロップダウンリスト付きのテキスト要素を表示します。ドロップダウンリストには コマンドが含まれ、ユーザーがクリックまたはタッチで呼び出すことができます。

Menu コントロールは**ComboBox** を継承するため、**ComboBox** と同様に、**Menu** コントロールに 項目を挿入したり、スタイルを設定することができます（**itemsSource** プロパティを参照）。

Menu コントロールには、ユーザーがメニューから項目を選択したときに発生する **itemClicked** イベントが追加されています。イベントハンドラは、 **Menu** コントロールを検査して、 クリックされた項目を特定できます。次に例を示します。

```
var menu = new wijmo.input.Menu(hostElement);
menu.header = 'Main Menu';
menu.itemsSource = ['option 1', 'option 2', 'option 3'];
menu.itemClicked.addHandler(function(sender, args) {
var menu = sender;
  alert('Thanks for selecting item ' + menu.selectedIndex + ' from menu ' + menu.header + '!');
});
```

以下の例は、**itemClicked** イベントを処理する メニューを作成する方法を示します。

サンプル

Show me (https://jsfiddle.net/Wijmo5/5fe93pm8)

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- collectionView
- command
- commandParameterPath
- commandPath
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- header
- headerPath
- hostElement
- inputElement
- isAnimated
- isButton
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable

- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- maxDropDownHeight
- maxDropDownWidth
- openOnHover
- owner
- placeholder
- rightToLeft
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- subItemsPath
- text

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- hide
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemClicked
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged

- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll
- show

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemClicked
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?): Menu
```

Menu クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **Menu**

プロパティ

- autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

command

項目がクリックされたときに実行されるコマンドを取得または設定します。

コマンドは以下の2つのメソッドを実装するオブジェクトです。

- **executeCommand(parameter)**: このメソッドはコマンドを実行します。
- **canExecuteCommand(parameter)**: このメソッドは、コントローラーがコマンドを実行できるかどうかを決定するブール値を返します。 このメソッドがfalseを返す場合、メニューオプションは無効になります。

また、**commandPath** プロパティを使用して個々の項目のコマンドを設定することもできます。

型 **any**

commandParameterPath

commandPath プロパティによって指定されたコマンドで使用するパラメーターを含むプロパティの名前を取得または設定します。

型 **string**

commandPath

ユーザーが項目をクリックしたときに実行されるコマンドを含むプロパティの名前を取得または設定します。

コマンドは以下の2つのメソッドを実装するオブジェクトです。

- **executeCommand(parameter)**: このメソッドはコマンドを実行します。
- **canExecuteCommand(parameter)**: このメソッドは、コントローラーがコマンドを実行できるかどうかを決定するブール値を返します。 このメソッドがfalseを返す場合、メニューオプションは無効になります。

型 **string**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

header

Menu 要素に表示されるHTMLテキストを取得または設定します。

型 **string**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 **ComboBox**
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **DropDown**
型 **HTMLInputElement**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

isButton

このMenu を通常のメニューではなく分割ボタンとして動作させるかどうかを 指定する値を取得または設定します。

通常のメニューと分割ボタンの違いは、ユーザーがメニューヘッダーを クリックしたときの動作です。 通常のメニューでは、ヘッダをクリックするとメニューオプションが表示されるか、非表示になります。 分割ボタンの場合は、ヘッダーをクリックすると、ドロップダウンリストから 項目を選択した場合と同様に、itemClicked イベントが発生したり、 最後に選択したオプションに関連するコマンドが呼び出されます。

メニュー項目のクリックと、分割ボタンのボタン部分のクリックを区別するには、 イベント送信元のisDroppedDown プロパティの値を確認します。 これがtrueの場合は、メニュー項目がクリックされました。 falseの場合は、ボタンがクリックされました。

たとえば、以下のコードは、デフォルトの項目/コマンドを変更するためにのみ ドロップダウンリストを使用し、ボタンがクリックされた場合にのみアクションを トリガする分割ボタンを実装します。

```
<-- view -->
<wj-menu is-button="true" header="Run" value="browser"
  item-clicked="itemClicked(s, e)">
  <wj-menu-item value="'Internet Explorer'">Internet Explorer</wj-menu-item>
  <wj-menu-item value="'Chrome'">Chrome</wj-menu-item>
  <wj-menu-item value="'FireFox'">FireFox</wj-menu-item>
  <wj-menu-item value="'Safari'">Safari</wj-menu-item>
  <wj-menu-item value="'Opera'">Opera</wj-menu-item>
</wj-menu>

// コントローラー
$scope.browser = 'Internet Explorer';
$scope.itemClicked = function (s, e) {

  // ドロップダウンでなかった場合は、ボタンがクリックされました
  if (!s.isDroppedDown) {
    alert('running ' + $scope.browser);
  }
}
```

型 **boolean**

isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **`ComboBox`** control, and to true on the **`AutoComplete`** control.

継承元 **`ComboBox`**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **`DropDown`**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **`ComboBox`**, **`InputDate`**, **`InputTime`**, **`InputDateTime`**, and **`InputColor`**. It defaults to false for the **`AutoComplete`** control.

継承元 **`DropDown`**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the items to select from.

Setting this property to an array causes the **`ComboBox`** to create an internal **`ICollectionView`** object that is exposed by the **`collectionView`** property.

The **`ComboBox`** selection is determined by the current item in its **`collectionView`**. By default, this is the first item in the collection. You may change this behavior by setting the **`currentItem`** property of the **`collectionView`** to null.

継承元 **ComboBox**
型 **any**

listBox

ドロップダウンに示されている **`ListBox`** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

- openOnHover

Gets or sets a value that determines whether the menu (and any sub-menus) should open and close automatically when the mouse hovers over the items.

The default value for this property is **false**.

型 **boolean**

- owner

Menu を所有する要素を取得または設定します。この変数は、単一のメニューはいくつかの要素のコンテキストメニューとして使用する場合wj-context-menuで設定します。

型 **HTMLElement**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValue

selectedValuePath を使用して取得された**selectedItem** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValuePath

selectedValue を**selectedItem** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **ListBox** should include group header items to delimit data groups.

Data groups are created by modifying the **groupDescriptions** property of the **ICollectionView** object used as an **itemsSource**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- subItemsPath

Gets or sets the name of the property that contains an array with items to be displayed in a sub-menu.

型 **string**

- text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hide

```
hide(): void
```

メニューを非表示にします。このメソッドは、**show** メソッドを使用して表示したコンテキストメニューを非表示にする場合に便利です。

戻り値 **void**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemClicked

```
onItemClicked(e?: EventArgs): void
```

itemClicked イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

show

```
show(position?: any): void
```

指定された場所にメニューを表示します。

このメソッドは、メニューをコンテキストメニューとして使用して、 ページ内のいくつかの要素にアタッチする場合に便利です。次に例を示します。

```
// メニューを作成します
var div = document.createElement('div');
var menu = new wijmo.input.Menu(div, {
    itemsSource: 'New,Open,Save,Exit'.split(','),
    itemClicked: function (s, e) {
        alert('thanks for picking ' + menu.selectedIndex);
    }
});

// メニューをいくつかの要素のコンテキストメニューとして使用します
var element = document.getElementById('btn');
element.addEventListener('contextmenu', function (e) {
    e.preventDefault();
    menu.show(e);
});
```

パラメーター

- **position**: **any** OPTIONAL
  メニューを表示する位置を指定するオプションの**MouseEvent**または参照要素。

  指定しない場合、メニューは画面の中心に表示されます。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemClicked

ユーザーがメニューから項目を選択したときに発生します。イベントハンドラで、イベント送信元の **selectedIndex** プロパティを調べてどの項目が選択されたかを確認できます。

引数 **EventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

MultiAutoComplete クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **AutoComplete**
派生クラス **WjMultiAutoComplete**
表示 ☐継承されたメンバー ☐イベント発生元

MultiAutoComplete コントロールを使用すると、カスタムオブジェクトや 単純な文字列を含むリストから項目を選択できます。

次の例では、**MultiAutoComplete** を使用して単一のリストから選択された複数項目を入力する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/94c6wb77)

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- collectionView
- controlTemplate
- cssMatch
- delay
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- headerPath
- hostElement
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceFunction
- listBox
- maxDropDownHeight
- maxDropDownWidth
- maxItems
- maxSelectedItems
- minLength
- placeholder

- rightToLeft
- searchMemberPath
- selectedIndex
- selectedItem
- selectedItems
- selectedMemberPath
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- text

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onSelectedItemsChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged

- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- selectedItemsChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?): MultiAutoComplete
```

MultiAutoComplete クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **MultiAutoComplete**

プロパティ

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

cssMatch

検索語と一致するすべての部分をコンテンツで強調表示するために使用する CSSクラスの名前を取得または設定します。

継承元 **AutoComplete**
型 **string**

delay

キーストロークが発生してから検索が実行されるまでの遅延（ミリ秒単位）を取得または設定します。

The default value for this property is **500** milliseconds.

継承元 **AutoComplete**
型 **number**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 **ComboBox**
型 **string**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **DropDown**
型 **HTMLInputElement**

- isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **ComboBox** control, and to true on the **AutoComplete** control.

継承元 **ComboBox**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

itemsSourceFunction

ユーザーの入力に従ってリスト項目を動的に提供する関数を取得または設定します。

この関数は以下の3つのパラメーターをとります。

- ユーザーが入力したクエリー文字列
- 返す項目の最大数
- 結果が取得されたときに呼び出すコールバック関数

例:

```
autoComplete.itemsSourceFunction = function (query, max, callback) {

  // サーバーから結果を取得します。
  var params = { query: query, max: max };
  $.getJSON('companycatalog.ashx', params, function (response) {

    // コントロールに結果を返します。
    callback(response);
  });
};
```

継承元 **AutoComplete**
型 **Function**

listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

maxItems

ドロップダウンリストに表示する項目の最大数を取得または設定します。

The default value for this property is **6**.

継承元 **AutoComplete**
型 **number**

maxSelectedItems

選択できる最大の項目数を取得または設定します。このプロパティをnull（デフォルト値）に設定すると、ユーザーは項目をいくつでも選択できます。

型 **number**

minLength

オートコンプリート候補を検索するために必要な入力の最小長さを取得または設定します。

The default value for this property is **2**.

継承元 **AutoComplete**
型 **number**

placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

searchMemberPath

項目の検索時に使用するプロパティのカンマ区切りリストを含む文字列を取得または設定します。

デフォルトでは、**AutoComplete** コントロールは、**displayMemberPath** プロパティで指定された プロパティと比較して一致を検索します。**searchMemberPath** プロパティを使用すると、 追加のプロパティを使用して検索を行うことができます。

たとえば、以下のコードは、会社名を表示し、 会社名、シンボル、および国に基づいて検索を行います。

```
var ac = new wijmo.input.AutoComplete('#autoComplete', {
  itemsSource: companies,
  displayMemberPath: 'name',
  searchMemberPath: 'symbol,country'
});
```

継承元 **AutoComplete**
型 **string**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedItems

現在選択されている項目が含まれる配列を取得または設定します。

型 **any**[]

- selectedMemberPath

どの項目が選択されるかの制御に使用されるプロパティの名前を取得または設定します。

型 **string**

- selectedValue

`selectedValuePath` を使用して取得された**`selectedItem`** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValuePath

`selectedValue` を**`selectedItem`** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

- showDropDownButton

コントロールでドロップダウンボタンを表示しないことを示す値をオーバーライドします。

型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **`ListBox`** should include group header items to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as an **`itemsSource`**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onSelectedItemsChanged

```
onSelectedItemsChanged(e?: EventArgs): void
```

selectedItemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

selectedItemsChanged

selectedItems プロパティの値が変化すると発生します。

引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

MultiSelect クラス

ファイル wijmo.input.js
モジュール **wijmo.input**
基本クラス **ComboBox**
派生クラス **WjMultiSelect**
表示 ☐継承されたメンバー ☐イベント発生元

MultiSelect コントロールを使用すると、カスタムオブジェクトや単純な文字列を 含むドロップダウンリストから複数の項目を選択できます。

MultiSelect コントロールは**ComboBox** を拡張し、 **itemsSource** 、**displayMemberPath** などの 通常のプロパティをすべて含みます。

また、**ListBox** コントロールと同様に、項目がオンになっているかどうかを 特定するためのプロパティの名前を定義する**checkedMemberPath** プロパティを持ちます。

現在オンの（選択されている）項目は、 **checkedItems** プロパティを使用して取得できます。

コントロールヘッダーは自由にカスタマイズできます。デフォルトでは、コントロールヘッダーには、 選択されている項目が2つまで表示され、その後に項目数が表示されます。表示する最大項目数（**maxHeaderItems** ）、 選択項目がない場合に表示されるメッセージ（**placeholder** ）、 および項目数の表示に使用する書式文字列（**headerFormat** ） を変更できます。

アプリケーションに必要な条件に基づいてヘッダーコンテンツを生成する関数を 提供することもできます（**headerFormatter** ）。

次の例では、**MultiSelect** コントロールを使用してドロップダウンリストから複数の項目を選択する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/44w7fob2)

コンストラクタ

- constructor

プロパティ

- autoExpandSelection
- checkedItems
- checkedMemberPath
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- headerFormat
- headerFormatter
- headerPath
- hostElement
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching

- isUpdating
- itemFormatter
- itemsSource
- listBox
- maxDropDownHeight
- maxDropDownWidth
- maxHeaderItems
- placeholder
- rightToLeft
- selectAllLabel
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- showSelectAllCheckbox
- text

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onCheckedItemsChanged
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener

- selectAll

イベント

- checkedItemsChanged
- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?): MultiSelect
```

MultiSelect クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **MultiSelect**

プロパティ

- autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- checkedItems

現在チェックされている項目を含む配列を取得または設定します。

型 **any[]**

- checkedMemberPath

各項目の隣に配置したチェックボックスを制限するために使用されるプロパティの名前を取得または設定します。

型 **string**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

headerFormat

maxHeaderItems より以上の項目がチェックされた場合、ヘッダー コンテンツを作成するために使用する書式文字列を取得または設定します。書式文字列は、'{count}'の置換文字列が含まれて、現在にチェックされた項目数と置き換えます。英語カルチャでは、このプロパティのデフォルト値は'{count:n0} items selected'です。

型 **string**

headerFormatter

コントロールのヘッダにHTMLを得る関数を取得または設定します。

デフォルトでは、コントロールのヘッダの内容は**`placeholder`**、**`maxHeaderItems`**、および現在の選択に 基づいて決定します。

アプリケーションの基準に基づいて、カスタム文字列を返す関数を指定すると、ヘッダ内容のカスタマイズができます。

型 **Function**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 **`ComboBox`**
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **`Control`**
型 **HTMLElement**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **`DropDown`**
型 **HTMLInputElement**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **`DropDown`**
型 **boolean**

isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 **`ComboBox`**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **ComboBox** control, and to true on the **AutoComplete** control.

継承元 **ComboBox**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

- maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

- maxHeaderItems

コントロールのヘッダに表示する項目の最大数を取得または設定します。

項目が選択されていない場合、ヘッダが、**placeholder** プロパティで 指定されたテキストが表示されます。

選択された項目の数は、**maxHeaderItems** プロパティの値より以下に ある場合、選択された項目はヘッダに示します。

選択された項目の数は、**maxHeaderItems** プロパティの値より大きい場合、 ヘッダは選択された項目ではなく、項目数表示します。

型 **number**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectAllLabel

showSelectAllCheckbox プロパティがtrueに設定されている場合、表示される「すべて選択」チェックボックスのラベルとして使用する文字列を取得または設定します。

このプロパティはデフォルトでnullに設定されます。これにより、コントロールには、ローカライズ化された文字列 「すべて選択」が表示されます。

型 **string**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValue

selectedValuePath を使用して取得された**selectedItem** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValuePath

selectedValue を**selectedItem** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **ListBox** should include group header items to delimit data groups.

Data groups are created by modifying the **groupDescriptions** property of the **ICollectionView** object used as an **itemsSource**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- showSelectAllCheckbox

コントロールのすべての項目を選択または選択解除するために、項目の上に「すべて選択」チェックボックスを表示するかどうかを取得または設定します。

The default value for this property is **false**.

型 **boolean**

- text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

checkedItemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

checkedItemsChanged

checkedItems プロパティの値が変更されたときに発生します。

引数 **EventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

Popup クラス

ファイル	wijmo.input.js
モジュール	**wijmo.input**
基本クラス	**Control**
派生クラス	**DetailDialog**, **WjPopup**
表示	☐継承されたメンバー ☐イベント発生元

要素をポップアップとして表示するクラス。

ポップアップは**owner** 要素を持つことができます。その場合、ポップアップは、 **showTrigger** プロパティと**hideTrigger** プロパティによって 指定されるアクションに基づいて表示または非表示にできるリッチツールチップとして動作します。

オーナー要素を持たないポップアップはダイアログのように動作します。画面の中央に配置され、 **show** メソッドを使用して表示されます。

Popup を閉じるには、**hide** メソッドを呼び出します。

または、**Popup** 内に'wj-hide'文字列で始まるクラスを持つクリック可能な 要素がある場合は、その要素がクリックされると**Popup** は非表示になります。さらに、そのクラス名に**dialogResult** プロパティが設定され、 呼び出し元が適切なアクションを実行できます。

たとえば、次の**Popup** は、ユーザーが［OK］または［Cancel］ボタンを クリックすると非表示になり、**dialogResult** プロパティが 'wj-hide-cancel'または'wj-hide-ok'のいずれかに設定されます。

```
<button id="btnPopup">Show Popup</button>
<wj-popup owner="#btnPopup" style="padding:12px">
  <p>下のいずれかのボタンを押して、ポップアップを非表示にします。</p>
  <hr/>
  <button class="wj-hide-ok" ng-click="handleOK()">OK</button>
  <button class="wj-hide-cancel">キャンセル</button>
</wj-popup>
```

次の例では、**Popup** コントロールを使用して所有者の要素とダイアログにアタッチされているポップアップを実装する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/j9t6s1xp)

コンストラクタ

- constructor

プロパティ

- content
- dialogResult
- dialogResultEnter
- fadeIn
- fadeOut
- hideTrigger
- hostElement
- isDisabled
- isDraggable
- isTouching
- isUpdating
- isVisible
- modal
- owner
- removeOnHide
- rightToLeft

- showTrigger

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hide
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onHidden
- onHiding
- onLostFocus
- onRefreshed
- onRefreshing
- onShowing
- onShown
- refresh
- refreshAll
- removeEventListener
- show

イベント

- gotFocus
- hidden
- hiding
- lostFocus
- refreshed
- refreshing
- showing
- shown

コンストラクタ

constructor

```
constructor(element: any, options?: any): Popup
```

Popup クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **Popup**

プロパティ

content

このPopup に含まれるHTML要素を取得または設定します。

型 **HTMLElement**

dialogResult

Popup を非表示にした後に、そのコンテンツを処理するために使用できる値を取得または設定します。

このプロパティは、**Popup** が表示されたときにnullに設定され、ボタンクリックイベントに 応答して、または**hide** メソッドの呼び出しの中で設定できます。

型 **any**

dialogResultEnter

Popup 表示中にユーザーが［Enter］キーを押したときに**dialogResult** として 使用される値を取得または設定します。

ユーザーが［Enter］を押し、**dialogResultEnter** プロパティがnullでない場合、 ポップアップは、すべての子要素が有効な状態かどうかをチェックします。 有効な状態の場合は、ポップアップが閉じ、 **dialogResult** プロパティが **dialogResultEnter** プロパティの値に設定されます。

型 **any**

fadeIn

Popup を表示するときにフェードアウトアニメーションを使用する かどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

fadeOut

Popup を非表示にするときにフェードアウトアニメーションを使用する かどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

hideTrigger

ポップアップを非表示にするアクションを取得または設定します。

デフォルトでは、**showTrigger** プロパティは**Blur** に設定されており、 フォーカスを失った場合、またはEscキーを押した時にポップアップが非表示になります。

hideTrigger プロパティを**Click** に設定すると、ポップアップはオーナ要素を クリックする場合、またはEscキーを押した時非表示になります。

hideTrigger プロパティを**None** に設定すると、Popupは**hide** メソッドを 呼び出します。

型 **PopupTrigger**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isDraggable

ポップアップをそのヘッダーでマウスによってドラッグできるかどうかを示す値を取得または設定します。

ヘッダーは '.wj-dialog-header' のCSSセレクターによって識別されます。 ダイアログに 'wj-dialog-header'クラスの要素が含まれていない場合、ユーザーはポップアップをドラッグできません。

ポップアップをドラッグ可能にする場合は、 '.wj-dialog-header' CSSセレクタのcursorプロパティを設定することができます。次に例を示しています。

```
<style>
  .wj-popup {
    width: 30%;
  }
  .wj-dialog-header {
    cursor: move;
  }
</style>
```

The default value for this property is **false**.

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isVisible

Popupが現在表示されているかどうかを決定する値を取得します。

型 **boolean**

modal

Popup をモーダルダイアログとして表示するかどうかを決定する値を 取得または設定します。

モーダルダイアログは、 **Popup** がページ内の他のコンテンツより目立つように、 暗い背景が付けられます。

ダイアログを完全にモーダルにするには、 **hideTrigger** プロパティを **None** に設定して、ユーザーが背景をクリックしても ダイアログを閉じることができないようにします。この場合は、**hide** メソッドが呼び出されるか、 ユーザーが［Esc］キーを押した場合にのみ、ダイアログが閉じられます。

The default value for this property is **false**.

型 **boolean**

owner

Popup を所有する要素を取得または設定します。オーナーがnullの場合、ポップアップはダイアログとして動作します。ダイアログは、画面の中央に配置され、**show** メソッドを使用して表示する必要があります。

型 **HTMLElement**

removeOnHide

Popup が非表示になっているときに**Popup** 要素をDOMから削除するか非表示にするかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

showTrigger

Popup を表示するアクションを取得または設定します。デフォルトでは、 **showTrigger** プロパティは**Click** に設定されており、オーナー要素をクリックすると、ポップアップが表示されます。 **showTrigger** プロパティを**None** に設定すると、ポップアップは**show** メソッドを呼び出すのみで表示されます。

型 **PopupTrigger**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能性があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hide

```
hide(dialogResult?: any): void
```

Popup を非表示にします。

パラメーター

- **dialogResult**: **any** OPTIONAL
  Popup を閉じる前に**dialogResult** プロパティに割り当てられる オプションの値。

戻り値 **void**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onHidden

```
onHidden(e?: EventArgs): void
```

hidden イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onHiding

```
onHiding(e: CancelEventArgs): boolean
```

hiding イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onShowing

```
onShowing(e: CancelEventArgs): boolean
```

showing イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

戻り値 **boolean**

onShown

```
onShown(e?: EventArgs): void
```

shown イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

show

```
show(modal?: boolean, handleResult?: Function): void
```

Popup を表示します。

パラメーター

- **modal**: **boolean** OPTIONAL
  ポップアップをモーダルダイアログとして表示するかどうか。 指定した場合、その値が **modalプロパティに設定されます**。
- **handleResult**: **Function** OPTIONAL
  ポップアップが非表示になると、コールバックが呼び出されます。 コールバックが指定されている場合は、パラメータとしてポップアップを受け取ります。

  handleResultコールバックを使用すると、**hidden** イベントにハンドラを アタッチしなくても、呼び出し元がモーダルダイアログの結果を処理できます。 たとえば、 以下のコードは、**CollectionView** 内の現在の項目を編集するための ダイアログを表示します。 編集結果は、**dialogResult** 値に応じてコミット またはキャンセルされます。 次に例を示します。

```
$scope.editCurrentItem = function () {
  $scope.data.editItem($scope.data.currentItem);
  $scope.itemEditor.show(true, function (e) {
    if (e.dialogResult == 'wj-hide-ok') {
      $scope.data.commitEdit();
    } else {
      $scope.data.cancelEdit();
    }
  });
}
```

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

hidden

Popup が非表示になる後に発生します。

引数 **EventArgs**

hiding

Popup が非表示になる前に発生します。

引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

showing

Popup が表示される前に発生させます。

引数 **CancelEventArgs**

shown

ポップアップが表示された後に発生させます。

引数 **EventArgs**

DateSelectionMode 列挙体

ファイル `wijmo.input.js`
モジュール **wijmo.input**

日付選択動作を定義する定数を指定します。

メンバー

名前	値	説明
None	0	ユーザーは、マウスやキーボードで現在の値を変更できません。
Day	1	ユーザーは、日を選択できます。
Month	2	ユーザーは、月を選択できます。

PopupTrigger 列挙体

ファイル `wijmo.input.js`
モジュール **wijmo.input**

Popup コントロールの表示/非表示をトリガーするアクションを指定します。

メンバー

名前	値	説明
None	0	トリガーなし。コードを使用してポップアップの表示/非表示を実行する必要があります。
Click	1	オーナー要素がクリックされたときに、ポップアップを表示または非表示にします。
Blur	2	フォーカスを失ったときにポップアップが非表示になります。
ClickOrBlur	Click | Blur	オーナー要素がクリックされたときに、ポップアップを表示または非表示にします。フォーカスを失ったときは非表示にします。

wijmo.chart モジュール

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

FlexChart コントロールとそのコントロールに関連付けられたクラスを定義します。

この例では、**FlexChart** コントロールを作成し、それをデータ配列に連結しています。 このチャートには3つの系列があり、それぞれがソース配列に含まれるオブジェクトの 1つのプロパティに対応しています。

この例の最後の系列は、**chartType** プロパティを使用して、 他の系列に使用されているデフォルトのチャートタイプを オーバーライドします。

サンプル

Show me (https://jsfiddle.net/Wijmo5/6GB66)

クラス

- Axis
- AxisCollection
- ChartTooltip
- DataLabel
- DataLabelBase
- DataLabelRenderEventArgs
- DataPoint
- FlexChart
- FlexChartBase
- FlexChartCore
- FlexPie
- HitTestInfo
- Legend
- LineMarker
- Palettes
- PieDataLabel
- PlotArea
- PlotAreaCollection
- RenderEventArgs
- Series
- SeriesBase
- SeriesEventArgs
- SeriesRenderingEventArgs

インターフェイス

- IRenderEngine

列挙体

- AxisType
- ChartElement
- ChartType
- ImageFormat
- LabelPosition
- LineMarkerAlignment
- LineMarkerInteraction
- LineMarkerLines
- Marker
- OverlappingLabels
- PieLabelPosition
- Position
- SelectionMode
- SeriesVisibility
- Stacking
- TickMark

Axis クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
派生クラス **FlexRadarAxis**, **WjFlexChartAxis**

チャートの軸を表します。

コンストラクタ

- constructor

プロパティ

- actualMax
- actualMin
- axisLine
- axisType
- binding
- format
- hostElement
- itemFormatter
- itemsSource
- labelAlign
- labelAngle
- labelPadding
- labels
- logBase
- majorGrid
- majorTickMarks
- majorUnit
- max
- min
- minorGrid
- minorTickMarks
- minorUnit
- name
- origin
- overlappingLabels
- plotArea
- position
- reversed
- title

メソッド

- convert
- convertBack
- onRangeChanged

イベント

- rangeChanged

コンストラクタ

constructor

```
constructor(position?: Position): Axis
```

Axis クラスの新しいインスタンスを初期化します。

パラメーター

- **position**: **Position** OPTIONAL
  チャート上での軸の位置。

戻り値 **Axis**

プロパティ

actualMax

実際の軸の最大を取得します。

これは、数値またはDateオブジェクト（時間ベースのデータの場合）を返します。

型 **any**

actualMin

実際の軸の最小を取得します。

これは、数値またはDateオブジェクト（時間ベースのデータの場合）を返します。

型 **any**

axisLine

軸線が表示されるかどうかを示す値を取得または設定します。 The default value for this property is **true**.

型 **boolean**

axisType

軸タイプを取得します。

型 **AxisType**

binding

軸ラベルで使用する**itemsSource** プロパティの カンマ区切りのプロパティ名を取得または設定します。

最初の名前は軸の値を指定し、2番目は対応する軸ラベルを表します。デフォルト値は'value,text'です。

型 **string**

format

軸ラベルに使用される書式文字列を取得または設定します （**Globalize** を参照）。

型 **string**

hostElement

軸のホスト要素を取得します。

型 **SVGGElement**

itemFormatter

軸ラベルのitemFormatter関数を取得または設定します。

指定された場合、関数は次の2つのパラメータをとります。

- **render engine**：ラベルの書式設定に使用される**IRenderEngine** オブジェクト。
- **current label**：以下のプロパティを含むオブジェクト。
  - **value**：書式設定する軸ラベルの値。
  - **text**：ラベルに使用するテキスト。
  - **pos**：ラベルがレンダリングされる コントロール座標内の位置。
  - **cls**：ラベルに適用されるCSSクラス。

この関数は、プロパティが変更されるラベルの ラベルパラメータを返します。

次に例を示します。

```
chart.axisY.itemFormatter = function(engine, label) {
    if (label.val > 5){
        engine.textFill = 'red'; // 赤色のテキスト
        label.cls = null; // デフォルトのCSSなし
     }
    return label;
}
```

型 **Function**

itemsSource

軸ラベルの項目ソースを取得または設定します。

プロパティの名前は、**binding** プロパティによって指定されます。

次に例を示します。

```
// Axis.bindingのデフォルト値は'value,text'です
 chart.axisX.itemsSource = [ { value:1, text:'one' }, { value:2, text:'two' } ];
```

<

型 **any**

labelAlign

ラベルの配置を取得または設定します。

デフォルトでは、ラベルは中央に配置されます。サポートされている値は、'left'および'right'（x軸の場合）と'top'および'bottom'（y軸の場合）です。

型 **string**

labelAngle

軸ラベルの回転角度を取得または設定します。

角度は度単位で測定されます。有効な値の範囲は-90〜90です。

型 **number**

labelPadding

ラベルのパディングを取得または設定します。 The default value for this property is **5** pixels.

型 **number**

labels

軸ラベルを表示すかどうかを示す値を取得または設定します。 The default value for this property is **true**.

型 **boolean**

logBase

軸の対数の底を取得または設定します。

底が指定されていない場合、その軸は線形スケールになります。

`logBase` プロパティを使用すると、原点の周囲に集まっているデータが広がります。これは一部の金融データセットや経済データセットでよく見られます。

型 **number**

majorGrid

軸のグリッド線が表示されるかどうかを示す値を取得または設定します。

型 **boolean**

majorTickMarks

軸の目盛りマークの場所を取得または設定します。

型 **`TickMark`**

majorUnit

軸ラベル間の単位数を取得または設定します。

軸の値が日付の場合、単位は日数になります。

型 **number**

- max

軸に表示される最大値を取得または設定します。

値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

The default value for this property is **null**, which causes the chart to calculate the maximum value based on the data.

型 **any**

- min

軸に表示される最小値を取得または設定します。

値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

The default value for this property is **null**, which causes the chart to calculate the minimum value based on the data.

型 **any**

- minorGrid

軸の副グリッド線が表示されるかどうかを示す値を取得または設定します。

型 **boolean**

- minorTickMarks

小軸目盛りマークの場所を取得または設定します。

型 **`TickMark`**

- minorUnit

軸の補助目盛間の単位数を取得または設定します。

軸の値が日付の場合、単位は日数になります。

型 **number**

- name

軸の名前を取得または設定します。

型 **string**

- origin

軸が直交軸と交差する位置の値を取得または設定します。

型 **number**

- overlappingLabels

重なった軸ラベルの処理方法を示す値を取得または設定します。 The default value for this property is **OverlappingLabels.Auto**.

型 **`OverlappingLabels`**

plotArea

軸のプロットエリアを取得または設定します。

型 **PlotArea**

position

プロットエリアに対する軸の位置を取得または設定します。

型 **Position**

reversed

軸を反転させる（上から下方向、または右から左方向にする）かどうかを示す値を取得または設定します。 The default value for this property is **false**.

型 **boolean**

title

軸の横に表示されるタイトルのテキストを取得または設定します。

型 **string**

メソッド

convert

```
convert(val: number, maxValue?: number, minValue?: number): number
```

指定した値をデータ座標からピクセル座標に変換します。

パラメーター

- **val**: **number**
  変換するデータ値。
- **maxValue**: **number** OPTIONAL
  データの最大値（オプション）。
- **minValue**: **number** OPTIONAL
  データの最小値（オプション）。

戻り値 **number**

convertBack

```
convertBack(val: number): number
```

指定した値をピクセル座標からデータ座標に変換します。

パラメーター

- **val**: **number**
  変換し戻すピクセル座標。

戻り値 **number**

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

rangeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

イベント

rangeChanged

軸範囲が変更されたときに発生します。

引数 **EventArgs**

AxisCollection クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **ObservableArray**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart コントロールの**Axis** オブジェクトのコレクションを表します。

コンストラクタ

- constructor

プロパティ

- isUpdating

メソッド

- beginUpdate
- clear
- deferUpdate
- endUpdate
- getAxis
- implementsInterface
- indexOf
- insert
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

イベント

- collectionChanged

コンストラクタ

constructor

```
constructor(data?: any[]): ObservableArray
```

ObservableArray クラスの新しいインスタンスを初期化します。

パラメーター

- **data**: **any**[] OPTIONAL
  ObservableArray に格納する項目を含む配列。

継承元 **ObservableArray**
戻り値 **ObservableArray**

プロパティ

isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

継承元 **ObservableArray**
型

メソッド

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **ObservableArray**
戻り値 **void**

clear

```
clear(): void
```

配列からすべての項目を削除します。

継承元 **ObservableArray**
戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

継承元 **ObservableArray**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **ObservableArray**
戻り値 **void**

getAxis

```
getAxis(name: string): Axis
```

名前によって軸を取得します。

パラメーター

- **name**: **string**
  検索する軸の名前。

戻り値 **Axis**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **ObservableArray**
戻り値 **boolean**

indexOf

```
indexOf(name: string): number
```

名前によって軸のインデックスを取得します。

パラメーター

- **name**: **string**
  検索する軸の名前。

戻り値 **number**

insert

```
insert(index: number, item: any): void
```

配列内の指定した位置に項目を挿入します。

パラメーター

- **index**: **number**
  項目を追加する位置。

- **item**: **any**
  配列に追加する項目。

継承元 **ObservableArray**
戻り値 **void**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

collectionChanged イベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  変更の記述が含まれます。

継承元 **ObservableArray**
戻り値 **void**

push

```
push(...item: any[]): number
```

配列の末尾に1つ以上の項目を追加します。

パラメーター

- **...item**: **any[]**
  配列に追加する1つ以上の項目。

継承元 **ObservableArray**
戻り値 **number**

remove

```
remove(item: any): boolean
```

配列から項目を削除します。

パラメーター

- **item**: **any**
  削除する項目。

継承元 **ObservableArray**
戻り値 **boolean**

removeAt

```
removeAt(index: number): void
```

配列内の指定した位置にある項目を削除します。

パラメーター

- **index**: **number**
  削除する項目の位置。

継承元 **ObservableArray**
戻り値 **void**

setAt

```
setAt(index: number, item: any): void
```

配列内の指定した位置にある項目を設定します。

パラメーター

- **index**: **number**
  項目を設定する位置。

- **item**: **any**
  配列に設定する項目。

継承元 **ObservableArray**
戻り値 **void**

slice

```
slice(begin?: number, end?: number): any[]
```

配列の一部のシャローコピーを作成します。

パラメーター

- **begin**: **number** OPTIONAL
  コピーを開始する位置。

- **end**: **number** OPTIONAL
  コピーを終了する位置。

継承元 **ObservableArray**
戻り値 **any[]**

sort

```
sort(compareFn?: Function): this
```

配列の要素を適切な位置にソートします。

パラメーター

- **compareFn**: **Function** OPTIONAL
  ソート順序を定義する関数。 この関数は2つの引数を受け取り、-1（最初の引数の方が2番目の引数より小さい場合）、+1（大きい場合）、0（等しい場合）のいずれかの値を返す必要があります。 これを省略した場合は、各要素の文字列変換に従って辞書順にソートされます。

継承元 **ObservableArray**
戻り値 **this**

splice

```
splice(index: number, count: number, item?: any): any[]
```

配列からの項目の削除と配列への項目の追加の一方または両方を行います。

パラメーター

- **index**: **number**
  項目を追加または削除する位置。

- **count**: **number**
  配列から削除する項目の数。

- **item**: **any** OPTIONAL
  配列に追加する項目。

継承元 **ObservableArray**
戻り値 **any[]**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 **ObservableArray**
引数 **NotifyCollectionChangedEventArgs**

ChartTooltip クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **Tooltip**
表示 ☐継承されたメンバー ☐イベント発生元

Tooltip クラスを拡張してチャートのツールチップを提供します。

コンストラクタ

- constructor

プロパティ

- content
- gap
- hideDelay
- isContentHtml
- isVisible
- showAtMouse
- showDelay
- threshold

メソッド

- dispose
- getTooltip
- hide
- onPopup
- setTooltip
- show

イベント

- popup

コンストラクタ

constructor

```
constructor(): ChartTooltip
```

ChartTooltip クラスの新しいインスタンスを初期化します。

戻り値 **ChartTooltip**

プロパティ

content

ツールチップの内容を取得または設定します。

ツールチップの内容は文字列として指定するほかに、**HitTestInfo** オブジェクトをパラメーターとして受け取る関数として指定することもできます。

ツールチップの内容が文字列の場合、以下のパラメーターを含めることができます。

- **propertyName**: ポイントによって表されるデータオブジェクトのプロパティ。
- **seriesName**: データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**: データポイントのインデックス。
- **value**: データポイントの**Value**（**FlexChart** のyの値、または**FlexPie** の項目の値）。
- **x**: データポイントの**x**の値（FlexChartのみ）。
- **y**: データポイントの**y**の値（FlexChartのみ）。
- **name**: データポイントの**Name**（**FlexChart** のxの値、または**FlexPie** の凡例エントリ）。

パラメーターは波かっこで囲む必要があります。例:

```
// 'country'と'sales'はデータオブジェクトのプロパティです。
  chart.tooltip.content = '{country}, sales:{sales}';
```

次の例は、関数を使用してツールチップの内容を設定する方法を示します。

```
// ツールチップの内容を設定します。
  chart.tooltip.content = function (ht) {
    return ht.name + ":" + ht.value.toFixed();
  }
```

型 **any**

gap

ツールチップとターゲット要素との距離を取得または設定します。

The default value for the property is **6** pixels.

継承元 **Tooltip**
型 **number**

hideDelay

Gets or sets the delay, in milliseconds, before hiding the tooltip if the mouse remains over the element.

The default value for the property is **zero** milliseconds, which causes the tip to remain visible until the mouse moves away from the element.

継承元 **Tooltip**
型 **number**

isContentHtml

ツールチップのコンテンツをプレーンテキストとして表示するか、HTMLとして表示するかを決定する値を取得または設定します。

The default value for the property is **true**.

継承元 **Tooltip**
型 **boolean**

isVisible

ツールチップが表示されているかどうかを示す値を取得します。

継承元 **Tooltip**
型 **boolean**

showAtMouse

ツールチップをターゲット要素ではなくマウスの位置を基準にして配置するかどうかを決定する値を取得または設定します。

The default value for the property is **false**.

継承元 **Tooltip**
型 **boolean**

showDelay

マウスがターゲット要素に入ってからツールチップが表示されるまでの遅延（ミリ秒単位）を取得または設定します。

The default value for the property is **500** milliseconds.

継承元 **Tooltip**
型 **number**

threshold

ツールチップを表示する位置の要素からの最大距離を取得または設定します。

型 **number**

メソッド

dispose

```
dispose(): void
```

この**Tooltip** インスタンスに関連付けられたすべてのツールチップを削除します。

継承元 **Tooltip**
戻り値 **void**

getTooltip

```
getTooltip(element: any): string
```

指定した要素に関連付けられたツールチップの内容を取得します。

パラメーター

- **element**: **any**
  ツールチップで説明する要素、要素ID、またはコントロール。

継承元 **Tooltip**
戻り値 **string**

hide

```
hide(): void
```

ツールチップが現在表示されている場合、非表示にします。

継承元 **Tooltip**
戻り値 **void**

onPopup

```
onPopup(e: TooltipEventArgs): boolean
```

popup イベントを発生させます。

パラメーター

- **e**: **TooltipEventArgs**
  イベントデータを含む**TooltipEventArgs**。

継承元 **Tooltip**
戻り値 **boolean**

setTooltip

```
setTooltip(element: any, content: string): void
```

ページ上の指定した要素にツールチップの内容を割り当てます。

ページ上の任意の数の要素に対して同じツールチップを使用して情報を表示できます。要素からツールチップを削除するには、 **setTooltip** を呼び出して内容をnullに設定します。

パラメーター

- **element**: **any**
  ツールチップで説明する要素、要素ID、またはコントロール。

- **content**: **string**
  ツールチップの内容、またはツールチップの内容を含む要素のID。

継承元 **Tooltip**
戻り値 **void**

show

```
show(element: any, content: string, bounds?: Rect): void
```

指定した要素の横に、指定した内容を含むツールチップを表示します。

パラメーター

- **element**: **any**
  ツールチップで説明する要素、要素ID、またはコントロール。

- **content**: **string**
  ツールチップの内容、またはツールチップの内容を含む要素のID。

- **bounds**: **Rect** OPTIONAL
  （オプション）ツールチップが対象とする領域の範囲を定義する要素。 これを指定しない場合は、（**getBoundingClientRect**メソッドによって報告される）要素の範囲が使用されます。

継承元 **Tooltip**
戻り値 **void**

イベント

popup

ツールチップの内容が表示される前に発生します。

イベントハンドラでイベントパラメーターを変更してツールチップの内容をカスタマイズしたり、ツールチップの表示を抑制したりできます。

継承元 **Tooltip**
引数 **TooltipEventArgs**

DataLabel クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **DataLabelBase**
派生クラス **WjFlexChartDataLabel**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChartのポイントデータラベル。

プロパティ

- border
- connectingLine
- content
- offset
- position

メソッド

- onRendering

イベント

- rendering

プロパティ

border

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

継承元 **DataLabelBase**
型 **boolean**

connectingLine

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

継承元 **DataLabelBase**
型 **boolean**

content

データラベルの内容を取得または設定します。

この内容は文字列として指定するほかに、**HitTestInfo** オブジェクトをパラメーターとして受け取る関数として指定することもできます。

ラベルの内容が文字列の場合、以下のパラメーターを含めることができます。

- **seriesName**: データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**: データポイントのインデックス。
- **value**: データポイントの**Value**。
- **x**: データポイントの**x**の値（FlexChartのみ）。
- **y**: データポイントの**y**の値（FlexChartのみ）。
- **name**: データポイントの**Name**。
- **propertyName**: データオブジェクトのプロパティ。

パラメーターは波かっこで囲む必要があります（例: 'x={x}, y={y}'）。

以下の例では、データポイントのyの値をラベルに表示します。

```
// チャートを作成し、データポイントの上に配置したラベルにyのデータを表示します。
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

次の例は、関数を使用してデータラベルの内容を設定する方法を示します。

```
// データラベルの内容を設定します。
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

継承元 **DataLabelBase**
型 **any**

offset

ラベルからデータポイントまでのオフセットを取得または設定します。

継承元 **DataLabelBase**
型 **number**

position

データラベルの位置を取得または設定します。

型 **LabelPosition**

メソッド

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **DataLabelRenderEventArgs**
  ラベルのレンダリングに使用される**DataLabelRenderEventArgs** オブジェクト。

継承元 **DataLabelBase**
戻り値 **void**

イベント

rendering

データラベルをレンダリングする前に発生します。

継承元 **DataLabelBase**
引数 **DataLabelRenderEventArgs**

DataLabelBase クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
派生クラス **DataLabel**, **PieDataLabel**

DataLabel クラスと**PieDataLabel** クラスの基本抽象クラスを表します。

プロパティ

- border
- connectingLine
- content
- offset

メソッド

- onRendering

イベント

- rendering

プロパティ

border

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

型 **boolean**

connectingLine

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

型 **boolean**

content

データラベルの内容を取得または設定します。

この内容は文字列として指定するほかに、**HitTestInfo** オブジェクトをパラメーターとして受け取る関数として指定することもできます。

ラベルの内容が文字列の場合、以下のパラメーターを含めることができます。

- **seriesName**: データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**: データポイントのインデックス。
- **value**: データポイントの**Value**。
- **x**: データポイントの**x**の値（FlexChartのみ）。
- **y**: データポイントの**y**の値（FlexChartのみ）。
- **name**: データポイントの**Name**。
- **propertyName**: データオブジェクトのプロパティ。

パラメーターは波かっこで囲む必要があります（例: 'x={x}, y={y}'）。

以下の例では、データポイントのyの値をラベルに表示します。

```
// チャートを作成し、データポイントの上に配置したラベルにyのデータを表示します。
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

次の例は、関数を使用してデータラベルの内容を設定する方法を示します。

```
// データラベルの内容を設定します。
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

型 **any**

offset

ラベルからデータポイントまでのオフセットを取得または設定します。

型 **number**

メソッド

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **DataLabelRenderEventArgs**
  ラベルのレンダリングに使用される**DataLabelRenderEventArgs** オブジェクト。

戻り値 **void**

イベント

rendering

データラベルをレンダリングする前に発生します。

引数 **DataLabelRenderEventArgs**

DataLabelRenderEventArgs クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **RenderEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

DataLabel レンダリングイベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- empty
- engine
- hitTestInfo
- point
- text

コンストラクタ

constructor

```
constructor(engine: IRenderEngine, ht: HitTestInfo, pt: Point, text: string): DataLabelRenderEventArgs
```

DataLabelRenderEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **engine**: **IRenderEngine**
  （**IRenderEngine** ）使用するレンダリングエンジン。
- **ht**: **HitTestInfo**
  ヒットテスト情報。
- **pt**: **Point**
  参照ポイント。
- **text**: **string**
  ラベルテキスト。

戻り値 **DataLabelRenderEventArgs**

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

型 **boolean**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

engine

チャート要素のレンダリングに使用する**IRenderEngine** オブジェクトを取得します。

継承元 **RenderEventArgs**
型 **IRenderEngine**

hitTestInfo

ヒットテスト情報を取得します。

型 **HitTestInfo**

point

ラベルに関連付けられるコントロール座標内のポイントを取得します。

型 **Point**

text

ラベルのテキストを取得または設定します。

型 **string**

DataPoint クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
派生クラス **WjFlexChartDataPoint**

データポイントを（x座標とy座標で）表すクラス。

X座標とY座標は、数値またはDateオブジェクト（時間ベースのデータの場合）として指定できます。

コンストラクタ

- constructor

プロパティ

- x
- y

コンストラクタ

constructor

```
constructor(x?: any, y?: any): DataPoint
```

DataPoint クラスの新しいインスタンスを初期化します。

パラメーター

- **x**: **any** OPTIONAL
  新しいDataPointのX 座標。
- **y**: **any** OPTIONAL
  新しいDataPointのY 座標。

戻り値 **DataPoint**

プロパティ

x

このDataPoint のX座標値を取得または設定します。

型 **any**

y

このDataPoint のY座標値を取得または設定します。

型 **any**

FlexChart クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **FlexChartCore**
派生クラス **WjFlexChart**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart コントロールは、データを視覚化する強力で柔軟な方法を提供します。

FlexChart コントロールを使用して、棒グラフ、折れ線グラフ、シンボルチャート、 バブルチャートなどのさまざまな形式でデータを表示するチャートを作成できます。

FlexChart コントロールを使用するには、**itemsSource** プロパティに データオブジェクトから成る配列を設定してから、1つ以上の**Series** オブジェクトを **series** プロパティに追加します。

chartType プロパティを使用して、すべての系列でデフォルトとして 使用される **ChartType** を定義します。**series** 配列のメンバの**chartType** プロパティを設定することで、系列ごとにチャートタイプを オーバーライドできます。

サンプル

Show me (https://jsfiddle.net/Wijmo5/6GB66)

コンストラクタ

- constructor

プロパティ

- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- interpolateNulls
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- options
- palette
- plotAreas
- plotMargin
- rightToLeft

- rotated
- selection
- selectionMode
- series
- stacking
- symbolSize
- tooltip

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged

seriesVisibilityChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FlexChart
```

FlexChart クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト

戻り値 **FlexChart**

プロパティ

axes

Axis オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

axisX

メインのX軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

axisY

メインのY軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

binding

Yの値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

bindingX

Xデータ値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

- chartType

作成するチャートのタイプを取得または設定します。 The default value for this property is **ChartType.Column**.

型 **ChartType**

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

- dataLabel

ポイントのデータラベルを取得または設定します。

継承元 **FlexChartCore**
型 **DataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 `FlexChartCore`
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

legendToggle

凡例項目をクリックしたときにチャート上の系列の表示/非表示を切り替えるかどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 `FlexChartCore`
型 **boolean**

options

さまざまなチャートオプションを取得または設定します。

以下のオプションがサポートされます。

bubble.maxSize：バブルチャートのシンボルの最大サイズ を指定します。デフォルト値は30ピクセルです。

bubble.minSize：バブルチャートのシンボルの最小サイズ を指定します。デフォルト値は5ピクセルです。

```
chart.options = {
  bubble: { minSize: 5, maxSize: 30 }
}
```

funnel.neckWidth：ファンネルグラフのネックの幅をパーセント値で指定します。 デフォルト値は0.2です。

funnel.neckHeight：ファンネルグラフのネックの高さをパーセント値で指定します。 デフォルト値は0です。

funnel.type：ファンネルグラフのタイプを指定します。これは、'rectangle'または'default'である必要があります。 タイプがrectangleに設定されている場合、neckWidthとneckHeightは機能しません。

```
chart.options = {
  funnel: { neckWidth: 0.3, neckHeight: 0.3, type: 'rectangle' }
}
```

groupWidth：縦棒グラフのグループ幅または 横棒グラフのグループ高さを指定します。グループ幅は、ピクセル単位 または有効なスペースに対するパーセント値で指定できます。デフォルト値は'70%'です。

```
chart.options = {
  groupWidth : 50; // 50ピクセル
}

chart.options = {
  groupWidth : '100%'; // 100%ピクセル
}
```

型 **any**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string**[]

plotAreas

PlotArea オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **PlotAreaCollection**

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- rotated

Xが垂直、Yが水平になるように軸を反転するかどうかを示す値を

取得または設定します。

The default value for this property is **false**.

型 **boolean**

- selection

選択されているチャート系列を取得または設定します。

継承元 **FlexChartCore**
型 **SeriesBase**

- selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

- series

Series オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

- stacking

系列オブジェクトを積み重ねるかどうかを決定する値と、積み重ねる場合はその方法を取得または設定します。 The default value for this property is **Stacking.None**.

型 **Stacking**

- symbolSize

この**FlexChart** のすべてのSeriesオブジェクトに使用されるシンボルのサイズを取得または設定します。

このプロパティは、各**Series** オブジェクトのsymbolSizeプロパティによってオーバーライドできます。

The default value for this property is **10** pixels.

継承元 **FlexChartCore**
型 **number**

tooltip

チャートの**Tooltip** オブジェクトを取得します。

ツールチップの内容は、次のパラメータを含むテンプレートを使用して生成されます。

- **propertyName**： このポイントによって表されるデータオブジェクトの任意のプロパティ。
- **seriesName**： データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**： データポイントのインデックス。
- **value**： データポイントの**値**（**FlexChart** の場合はy値、**FlexPie** の場合は項目値）。
- **x**： データポイントの**x**値（FlexChartのみ）。
- **y**： データポイントの**y**値（FlexChartのみ）。
- **name**： データポイントの**名前**（**FlexChart** の場合はx値、**FlexPie** の場合は凡例エントリ）。

テンプレートを変更するには、ツールチップのコンテンツプロパティに新しい値を割り当てます。 次に例を示します。

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

チャートのツールチップを無効にできます。それには、テンプレートを空の文字列に設定します。

tooltip プロパティを使用して、次のように、**showDelay** や**hideDelay** などの ツールチップパラメータをカスタマイズすることもできます。

```
chart.tooltip.showDelay = 1000;
```

詳細とオプションについては、**ChartTooltip** プロパティを参照してください。

継承元　　**FlexChartCore**
型　　**ChartTooltip**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Point をデータ座標からコントロール座標に変換します。

パラメーター

- **pt**: **any**
  データ座標の**Point** 、またはデータ座標のポイントのX座標。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

seriesVisibilityChanged イベントを発生させます。

パラメーター

- **e**: **SeriesEventArgs**
  イベントデータを含む**SeriesEventArgs** オブジェクト。

継承元 **FlexChartCore**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

pointToData

```
pointToData(pt: any, y?: number): Point
```

Point をコントロール座標からチャートデータ座標に変換します。

パラメーター

- **pt**: **any**
  変換するポイント（コントロール座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**`ImageFormat`** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **`FlexChartBase`**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **`FlexChartBase`**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

seriesVisibilityChanged

系列の表示/非表示が変更されたときに発生します。たとえば、legendToggleプロパティをtrueに設定したり、ユーザーが凡例をクリックしたときです。

継承元 **FlexChartCore**
引数 **SeriesEventArgs**

FlexChartBase クラス

ファイル	`wijmo.chart.js`
モジュール	**wijmo.chart**
基本クラス	**Control**
派生クラス	**FlexChartCore**, **FlexPie**, **TreeMap**
表示	☐ 継承されたメンバー ☐ イベント発生元

FlexChartおよびFlexPieの派生元の**FlexChartBase** コントロール。

コンストラクタ

- constructor

プロパティ

- collectionView
- footer
- footerStyle
- header
- headerStyle
- hostElement
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- palette
- plotMargin
- rightToLeft
- selectionMode

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered

- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged

コンストラクタ

constructor

```
constructor(element: any, options?, invalidateOnResize?: boolean): Control
```

Control クラスの新しいインスタンスを初期化してDOM要素にアタッチします。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。
- **invalidateOnResize**: **boolean** OPTIONAL
  コントロールのサイズが変更されたときにコントロールを無効にするかどうか。

継承元 **Control**
戻り値 **Control**

プロパティ

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

型 **ICollectionView**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

型 **string**

footerStyle

チャートのフッタスタイルを取得または設定します。

型 **any**

header

チャートのヘッダに表示されるテキストを取得または設定します。

型 **string**

headerStyle

チャートのヘッダスタイルを取得または設定します。

型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

型 **any**

legend

チャートの凡例を取得または設定します。

型 **Legend**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

型 **string**[]

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

型 **any**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

型 **SelectionMode**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**ImageFormat** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

引数 **EventArgs**

FlexChartCore クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **FlexChartBase**
派生クラス **FlexChart**, **FlexRadar**, **FinancialChart**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart 用の中核的なチャートコントロール。

コンストラクタ

- constructor

プロパティ

- axes
- axisX
- axisY
- binding
- bindingX
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- interpolateNulls
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- palette
- plotAreas
- plotMargin
- rightToLeft
- selection
- selectionMode
- series
- symbolSize
- tooltip

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- dataToPoint
- deferUpdate

- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FlexChartCore
```

FlexChart クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **FlexChartCore**

プロパティ

- axes

Axis オブジェクトのコレクションを取得します。

型 **ObservableArray**

- axisX

メインのX軸を取得または設定します。

型 **Axis**

- axisY

メインのY軸を取得または設定します。

型 **Axis**

- binding

Yの値を含むプロパティの名前を取得または設定します。

型 **string**

- bindingX

Xデータ値を含むプロパティの名前を取得または設定します。

型 **string**

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

dataLabel

ポイントのデータラベルを取得または設定します。

型 **DataLabel**

footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

legendToggle

凡例項目をクリックしたときにチャート上の系列の表示/非表示を切り替えるかどうかを示す値を取得または設定します。 The default value for this property is **false**.

型 **boolean**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string**[]

plotAreas

PlotArea オブジェクトのコレクションを取得します。

型 **PlotAreaCollection**

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selection

選択されているチャート系列を取得または設定します。

型 **SeriesBase**

selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

series

Series オブジェクトのコレクションを取得します。

型 **ObservableArray**

symbolSize

この**FlexChart** のすべてのSeriesオブジェクトに使用されるシンボルのサイズを取得または設定します。

このプロパティは、各**Series** オブジェクトのsymbolSizeプロパティによってオーバーライドできます。

The default value for this property is **10** pixels.

型 **number**

tooltip

チャートの**Tooltip** オブジェクトを取得します。

ツールチップの内容は、次のパラメータを含むテンプレートを使用して生成されます。

- **propertyName**： このポイントによって表されるデータオブジェクトの任意のプロパティ。
- **seriesName**： データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**： データポイントのインデックス。
- **value**： データポイントの**値**（**FlexChart** の場合はy値、**FlexPie** の場合は項目値）。
- **x**： データポイントの**x**値（FlexChartのみ）。
- **y**： データポイントの**y**値（FlexChartのみ）。
- **name**： データポイントの**名前**（**FlexChart** の場合はx値、**FlexPie** の場合は凡例エントリ）。

テンプレートを変更するには、ツールチップのコンテンツプロパティに新しい値を割り当てます。 次に例を示します。

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

チャートのツールチップを無効にできます。それには、テンプレートを空の文字列に設定します。

tooltip プロパティを使用して、次のように、**showDelay** や**hideDelay** などの ツールチップパラメータをカスタマイズすることもできます。

```
chart.tooltip.showDelay = 1000;
```

詳細とオプションについては、**ChartTooltip** プロパティを参照してください。

型 **ChartTooltip**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Point をデータ座標からコントロール座標に変換します。

パラメーター

- **pt**: **any**
  データ座標の**Point** 、またはデータ座標のポイントのX座標。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

戻り値 **Point**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。
- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

seriesVisibilityChanged イベントを発生させます。

パラメーター

- **e**: **SeriesEventArgs**
  イベントデータを含む**SeriesEventArgs** オブジェクト。

戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

pointToData

```
pointToData(pt: any, y?: number): Point
```

Point をコントロール座標からチャートデータ座標に変換します。

パラメーター

- **pt**: **any**
  変換するポイント（コントロール座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**`ImageFormat`** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **`FlexChartBase`**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **`FlexChartBase`**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

seriesVisibilityChanged

系列の表示/非表示が変更されたときに発生します。たとえば、legendToggleプロパティをtrueに設定したり、ユーザーが凡例をクリックしたときです。

引数 **SeriesEventArgs**

FlexPie クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **FlexChartBase**
派生クラス **Sunburst**, **WjFlexPie**
表示 ☐継承されたメンバー ☐イベント発生元

FlexPie コントロールは、選択可能なセグメントを持つ円グラフおよびドーナツグラフを提供します。

FlexPie コントロールを使用するには、**itemsSource** プロパティにデータから成る配列を設定し、**binding** と**bindingName** プロパティを使用して、項目値と項目名を含むプロパティを設定します。

コンストラクタ

- constructor

プロパティ

- binding
- bindingName
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- innerRadius
- isAnimated
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- offset
- palette
- plotMargin
- reversed
- rightToLeft
- selectedIndex
- selectedItemOffset
- selectedItemPosition
- selectionMode
- startAngle
- tooltip

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus

- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FlexPie
```

FlexPie クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  このコントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavascriptオブジェクト。

戻り値 **FlexPie**

プロパティ

- binding

チャート値を含むプロパティの名前を取得または設定します。

型 **string**

- bindingName

データ項目の名前を含むプロパティの名前を取得または設定します。

型 **string**

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

- dataLabel

ポイントのデータラベルを取得または設定します。

型 **PieDataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **`Control`**
型 **HTMLElement**

innerRadius

パイの内側半径のサイズを取得または設定します。

内側半径はパイ半径に対する割合として測定されます。

このプロパティのデフォルト値はゼロです（つまり、円グラフになります）。このプロパティをゼロより大きい値に設定すると、円グラフの中央に穴が開きます（これをドーナツグラフと呼びます）。

The default value for this property is **0**

型 **number**

isAnimated

項目が選択されたときにアニメーションを使用するかどうかを示す値を取得または設定します。

`selectedItemPosition` プロパティおよび**`selectionMode`** プロパティも参照してください。

The default value for this property is **false**.

型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **`Control`**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

offset

スライスのパイ中心からのオフセットを取得または設定します。

オフセットはパイ半径に対する割合として測定されます。

The default value for this property is **0**.

型 **number**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string**[]

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

- reversed

角度を反転（反時計回り）するかどうかを決定する値を取得または設定します。

デフォルト値はfalseです。この場合、角度は時計回りに測定されます。

The default value for this property is **false**.

型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectedIndex

選択されたスライスのインデックスを取得または設定します。

型 **number**

- selectedItemOffset

選択されたスライスのパイ中心からのオフセットを取得または設定します。

オフセットはパイ半径に対する割合として測定されます。

The default value for this property is **0**.

型 **number**

- selectedItemPosition

選択されたスライスの位置を取得または設定します。

このプロパティを'None'以外の値に設定すると、スライスを選択したときに円グラフが回転します。

円グラフをクリックしたときにスライスが選択されるようにするには、 **selectionMode** プロパティを'Point'に設定する必要があります。

The default value for this property is **Position.None**.

型 **Position**

- selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

startAngle

パイスライスの開始角度（度単位）を取得または設定します。

角度は9時の位置から時計回りに測定されます。

The default value for this property is **0**.

型 **number**

tooltip

チャートの**Tooltip** を取得します。

型 **ChartTooltip**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**ImageFormat** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **FlexChartBase**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **FlexChartBase**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

HitTestInfo クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**

指定されたページ座標にある**FlexChart** コントロールの部分に関する情報を格納します。

コンストラクタ

- constructor

プロパティ

- chart
- chartElement
- distance
- item
- point
- pointIndex
- series
- x
- y

コンストラクタ

constructor

```
constructor(chart: FlexChartBase, point: Point, element?: ChartElement): HitTestInfo
```

HitTestInfo クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartBase**
  チャートコントロール。
- **point**: **Point**
  元のポイント（ウィンドウ座標単位）。
- **element**: **ChartElement** OPTIONAL
  チャート要素。

戻り値 **HitTestInfo**

プロパティ

chart

このHitTestInfo を所有するFlexChartBase 。

型 **FlexChartBase**

chartElement

指定された座標にあるチャート要素を取得します。

型 **ChartElement**

- distance

最も近いデータポイントまでの距離を取得します。

型 **number**

- item

最も近いデータポイントに対応するデータオブジェクトを取得します。

型 **any**

- point

この**HitTestInfo** rが参照するコントロール座標内のポイントを取得します。

型 **Point**

- pointIndex

指定された座標にあるデータポイントのインデックスを取得します。

型 **number**

- series

指定された座標にあるチャート系列を取得します。

型 **SeriesBase**

- x

最も近いデータポイントのx値を取得します。

型 **any**

y

最も近いデータポイントのy値を取得します。

型 **any**

Legend クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
派生クラス **WjFlexChartLegend**

チャートの凡例を表します。

コンストラクタ

- constructor

プロパティ

- position
- title
- titleAlign

コンストラクタ

constructor

```
constructor(chart: FlexChartBase): Legend
```

Legend クラスの新しいインスタンスを初期化します。

パラメーター
- **chart**: **FlexChartBase**
  この**Legend** を所有する**FlexChartBase**。

戻り値 **Legend**

プロパティ

- position

凡例を表示するかどうか、表示する場合はプロットエリアに対してどの位置に表示するかを決定する値を取得または設定します。

型 **Position**

- title

凡例のタイトルを決定する値を取得または設定します。

型 **string**

- titleAlign

凡例の配置値を決定する値を取得または設定します。 有効な値は、"left" 、"center"または "right" です。

型 **string**

LineMarker クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
派生クラス **WjFlexChartLineMarker**

FlexChart用のLineMarkerの拡張機能を表します。

LineMarker は、データポイント値を表すテキスト領域と、 プロット領域全体に配置されるオプションの垂直または水平線（両方指定した場合は十字線）で構成されています。

interaction = Noneの場合は固定され、 interaction = Moveの場合はマウスまたはタッチ位置に従って移動します。 また、interaction = Dragの場合はユーザーが線をドラッグすると移動します。

例:

```
// yの値を示す、水平線の付いたインタラクティブマーカーを作成します。
  var lm = new wijmo.chart.LineMarker($scope.ctx.chart, {
      lines: wijmo.chart.LineMarkerLines.Horizontal,
      interaction: wijmo.chart.LineMarkerInteraction.Move,
      alignment : wijmo.chart.LineMarkerAlignment.Top
  });
  lm.content = function (ht) {

      // yの値を表示します。
      return lm.y.toFixed(2);
  }
```

コンストラクタ

- constructor

プロパティ

- alignment
- chart
- content
- dragContent
- dragLines
- dragThreshold
- horizontalPosition
- interaction
- isVisible
- lines
- seriesIndex
- verticalPosition
- x
- y

メソッド

- onPositionChanged
- remove

イベント

- positionChanged

コンストラクタ

constructor

```
constructor(chart: FlexChartCore, options?): LineMarker
```

LineMarker クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartCore**
  LineMarkerが表示されるチャート。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **LineMarker**

プロパティ

alignment

LineMarkerの内容の配置を取得または設定します。

デフォルトでは、LineMarkerはターゲットポイントの右下に表示されます。 '|'を使用してAlignmentの値を結合できます。

```
// 配置を左に設定します。
marker.alignment = wijmo.chart.LineMarkerAlignment.Left;

// 配置を左上に設定します。
marker.alignment = wijmo.chart.LineMarkerAlignment.Left | wijmo.chart.LineMarkerAlignment.Top;
```

型 **LineMarkerAlignment**

chart

LineMarkerを所有する**FlexChart** オブジェクトを取得します。

型 **FlexChartCore**

content

LineMarkerのテキストコンテンツをカスタマイズするためのコンテンツ関数を取得または設定します。

型 **Function**

dragContent

インタラクションモードが"Drag"の場合にマーカーの内容をドラッグできるかどうかを示す値を取得または設定します。

型 **boolean**

dragLines

インタラクションモードが"Drag"の場合に水平線または垂直線をドラッグしたときに両方の線が連動するかどうかを示す値を取得または設定します。

型 **boolean**

- dragThreshold

マーカーをドラッグできる水平線または垂直線からの最大距離を取得または設定します。

型 **number**

- horizontalPosition

プロットエリアに対するLineMarkerの水平位置を取得または設定します。

値の範囲は(0, 1)です。 値がnullまたは未定義で、**`interaction`** が wijmo.chart.LineMarkerInteraction.Moveまたは wijmo.chart.LineMarkerInteraction.Dragに設定されている場合、マーカーの水平位置はポインタの位置に基づいて自動的に 計算されます。

型 **number**

- interaction

LineMarkerのインタラクションモードを取得または設定します。

型 **`LineMarkerInteraction`**

- isVisible

LineMarkerの表示/非表示設定を取得または設定します。

型 **boolean**

- lines

LineMarkerの線の表示/非表示設定を取得または設定します。

型 **`LineMarkerLines`**

- seriesIndex

このLineMarkerが表示されるチャートの系列のインデックスを取得または設定します。 これは**`interaction`** プロパティが wijmo.chart.LineMarkerInteraction.Move またはwijmo.chart.LineMarkerInteraction.Dragに設定されているときに有効になります。

型 **number**

- verticalPosition

プロット領域に対するLineMarkerの垂直位置を取得または設定します。

値の範囲は(0, 1)です。 このプロパティの値がnullまたはundefinedで、**`interaction`** がwijmo.chart.LineMarkerInteraction.Moveまたは wijmo.chart.LineMarkerInteraction.Dragに設定されている場合、LineMarkerの垂直位置はポインタの位置に基づいて自動的に計算されます。

型 **number**

- x

現在のxの値をチャートのデータ座標として取得します。

型 **number**

y

現在のyの値をチャートのデータ座標として取得します。

型 **number**

メソッド

onPositionChanged

```
onPositionChanged(point: Point): void
```

positionChanged イベントを発生させます。

パラメーター

- **point**: **Point**
  LineMarkerを表示するターゲット位置。

戻り値 **void**

remove

```
remove(): void
```

チャートからLineMarkerを削除します。

戻り値 **void**

イベント

positionChanged

LineMarker の位置が変更された後に発生します。

引数 **Point**

Palettes クラス

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

これらはチャートの**Series** オブジェクト用の定義済みカラーパレットです。

カスタムカラーパレットを作成するには、文字列またはRGBA値の配列を指定します。

パレットは**FlexChart** コントロールおよび**FlexPie** コントロールに対して指定できます。 例:

```
chart.palette = Palettes.light;
```

以下の定義済みパレットが用意されています。

- standard (default)
- cocoa
- coral
- dark
- highcontrast
- light
- midnight
- modern
- organic
- slate
- zen
- cyborg
- superhero
- flatly
- darkly
- cerulan

PieDataLabel クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **DataLabelBase**
派生クラス **WjFlexPieDataLabel**
表示 ☐継承されたメンバー ☐イベント発生元

FlexPieのポイントデータラベル。

プロパティ

- border
- connectingLine
- content
- offset
- position

メソッド

- onRendering

イベント

- rendering

プロパティ

border

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

継承元 **DataLabelBase**
型 **boolean**

connectingLine

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

継承元 **DataLabelBase**
型 **boolean**

content

データラベルの内容を取得または設定します。

この内容は文字列として指定するほかに、**HitTestInfo** オブジェクトをパラメーターとして受け取る関数として指定することもできます。

ラベルの内容が文字列の場合、以下のパラメーターを含めることができます。

- **seriesName**: データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**: データポイントのインデックス。
- **value**: データポイントの**Value**。
- **x**: データポイントの**x**の値（FlexChartのみ）。
- **y**: データポイントの**y**の値（FlexChartのみ）。
- **name**: データポイントの**Name**。
- **propertyName**: データオブジェクトのプロパティ。

パラメーターは波かっこで囲む必要があります（例: 'x={x}, y={y}'）。

以下の例では、データポイントのyの値をラベルに表示します。

```
// チャートを作成し、データポイントの上に配置したラベルにyのデータを表示します。
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

次の例は、関数を使用してデータラベルの内容を設定する方法を示します。

```
// データラベルの内容を設定します。
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

継承元 **DataLabelBase**
型 **any**

offset

ラベルからデータポイントまでのオフセットを取得または設定します。

継承元 **DataLabelBase**
型 **number**

position

データラベルの位置を取得または設定します。

型 **PieLabelPosition**

メソッド

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **DataLabelRenderEventArgs**
  ラベルのレンダリングに使用される**DataLabelRenderEventArgs** オブジェクト。

継承元 **DataLabelBase**
戻り値 **void**

イベント

rendering

データラベルをレンダリングする前に発生します。

継承元 **DataLabelBase**
引数 **DataLabelRenderEventArgs**

PlotArea クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
派生クラス **WjFlexChartPlotArea**

チャートのプロットエリアを表します。

チャートには、複数の軸を持つ複数のプロットエリアを含めることができます。 プロットエリアに軸を割り当てるには、**Axis.plotArea**プロパティを使用します。次に例を示します。次に例を示します。

```
// プロットエリアを作成します
 var pa = new wijmo.chart.PlotArea();
 pa.row = 1;
 chart.plotAreas.push(pa);

 // 補助y軸を作成します
 var ay2 = new wijmo.chart.Axis(wijmo.chart.Position.Left);
 ay2.plotArea = pa; // 軸をプロットエリアにアタッチします
 chart.axes.push(ay2);

 // 最初の系列をy軸に沿ってプロットします
 chart.series[0].axisY = ay2;
```

コンストラクタ

- constructor

プロパティ

- column
- height
- name
- row
- style
- width

コンストラクタ

constructor

```
constructor(options?: any): PlotArea
```

PlotArea クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  プロットエリアの初期化オプション。

戻り値 **PlotArea**

プロパティ

column

プロットエリアの列インデックスを取得または設定します。 これにより、チャート上にプロットエリアの水平位置が決定されます。

型 **number**

- height

プロットエリアの高さを取得または設定します。

高さは、数値（ピクセル単位）または 書式'{number}*'の文字列（スターサイズ）で指定できます。

型 **any**

- name

プロットエリア名を取得または設定します。

型 **string**

- row

プロットエリアの行インデックスを取得または設定します。 これにより、チャート上にプロットエリアの垂直位置が決定されます。

型 **number**

- style

プロットエリアのスタイルを取得または設定します。

styleプロパティを使用して、プロットエリアの外観を設定できます。 次に例を示します。 次に例を示します。

```
pa.style = { fill: 'rgba(0,255,0,0.1)' };
```

型 **any**

- width

プロットエリアの幅を取得または設定します。

幅は、数値（ピクセル単位）または書式'{number}*'の文字列（スターサイズ）で指定できます。

型 **any**

PlotAreaCollection クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **ObservableArray**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChartCore コントロール内の**PlotArea** オブジェクトのコレクションを表します。

コンストラクタ

- constructor

プロパティ

- isUpdating

メソッド

- beginUpdate
- clear
- deferUpdate
- endUpdate
- getPlotArea
- implementsInterface
- indexOf
- insert
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

イベント

- collectionChanged

コンストラクタ

constructor

```
constructor(data?: any[]): ObservableArray
```

ObservableArray クラスの新しいインスタンスを初期化します。

パラメーター

- **data**: **any**[] OPTIONAL
  ObservableArray に格納する項目を含む配列。

継承元 **ObservableArray**
戻り値 **ObservableArray**

プロパティ

isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

継承元 **ObservableArray**
型

メソッド

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **ObservableArray**
戻り値 **void**

clear

```
clear(): void
```

配列からすべての項目を削除します。

継承元 **ObservableArray**
戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

継承元 **ObservableArray**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **ObservableArray**
戻り値 **void**

getPlotArea

```
getPlotArea(name: string): PlotArea
```

名前に基づいてプロットエリアを取得します。

パラメーター

- **name**: **string**
  検索するプロットエリアの名前。

戻り値 **PlotArea**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **ObservableArray**
戻り値 **boolean**

indexOf

```
indexOf(name: string): number
```

名前に基づいてプロットエリアのインデックスを取得します。

パラメーター

- **name**: **string**
  検索するプロットエリアの名前。

戻り値 **number**

insert

```
insert(index: number, item: any): void
```

配列内の指定した位置に項目を挿入します。

パラメーター

- **index**: **number**
  項目を追加する位置。

- **item**: **any**
  配列に追加する項目。

継承元 **ObservableArray**
戻り値 **void**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

collectionChanged イベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  変更の記述が含まれます。

継承元 **ObservableArray**
戻り値 **void**

push

```
push(...item: any[]): number
```

配列の末尾に1つ以上の項目を追加します。

パラメーター

- **...item**: **any[]**
  配列に追加する1つ以上の項目。

継承元 **ObservableArray**
戻り値 **number**

remove

```
remove(item: any): boolean
```

配列から項目を削除します。

パラメーター

- **item**: **any**
  削除する項目。

継承元 **ObservableArray**
戻り値 **boolean**

removeAt

```
removeAt(index: number): void
```

配列内の指定した位置にある項目を削除します。

パラメーター

- **index**: **number**
  削除する項目の位置。

継承元 **ObservableArray**
戻り値 **void**

setAt

```
setAt(index: number, item: any): void
```

配列内の指定した位置にある項目を設定します。

パラメーター

- **index**: **number**
  項目を設定する位置。

- **item**: **any**
  配列に設定する項目。

継承元 **ObservableArray**
戻り値 **void**

slice

```
slice(begin?: number, end?: number): any[]
```

配列の一部のシャローコピーを作成します。

パラメーター

- **begin**: **number** OPTIONAL
  コピーを開始する位置。

- **end**: **number** OPTIONAL
  コピーを終了する位置。

継承元 **ObservableArray**
戻り値 **any[]**

sort

```
sort(compareFn?: Function): this
```

配列の要素を適切な位置にソートします。

パラメーター

- **compareFn**: **Function** OPTIONAL
  ソート順序を定義する関数。 この関数は2つの引数を受け取り、-1（最初の引数の方が2番目の引数より小さい場合）、+1（大きい場合）、0（等しい場合）のいずれかの値を返す必要があります。 これを省略した場合は、各要素の文字列変換に従って辞書順にソートされます。

継承元 **ObservableArray**
戻り値 **this**

splice

```
splice(index: number, count: number, item?: any): any[]
```

配列からの項目の削除と配列への項目の追加の一方または両方を行います。

パラメーター

- **index**: **number**
  項目を追加または削除する位置。
- **count**: **number**
  配列から削除する項目の数。
- **item**: **any** OPTIONAL
  配列に追加する項目。

継承元 **ObservableArray**
戻り値 **any[]**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 **ObservableArray**
引数 **NotifyCollectionChangedEventArgs**

RenderEventArgs クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **CancelEventArgs**
派生クラス **DataLabelRenderEventArgs**, **SeriesRenderingEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

Series イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- empty
- engine

コンストラクタ

constructor

```
constructor(engine: IRenderEngine): RenderEventArgs
```

RenderEventArgs クラスの新しいインスタンスを初期化します。

パラメーター
- **engine**: **IRenderEngine**
  （**IRenderEngine** ）使用するレンダリングエンジン。

戻り値 **RenderEventArgs**

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

- engine

チャート要素のレンダリングに使用する**IRenderEngine** オブジェクトを取得します。

型 **IRenderEngine**

Series クラス

ファイル	`wijmo.chart.js`
モジュール	**wijmo.chart**
基本クラス	**SeriesBase**
派生クラス	**ErrorBar**, **WjFlexChartSeries**
表示	☐継承されたメンバー ☐イベント発生元

チャートに表示するデータポイントの系列を表します。

Series クラスは、基本的なチャートタイプのすべてをサポートします。**FlexChart** 系列コレクションに追加する**Series** オブジェクトごとに異なるチャートタイプを定義できます。**Series** オブジェクトに異なるチャートタイプを設定すると、チャートに設定された**chartType** プロパティ（系列コレクション内のすべての**Series** オブジェクトのデフォルト）がオーバーライドされます。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): SeriesBase
```

SeriesBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **SeriesBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

chartType

特定の系列のチャートタイプを取得または設定します。これはチャート全体に設定されたチャートタイプをオーバーライドします。 The default value for this property is **null**, which causes the series to use chart type defined by the parent chart.

型 **ChartType**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

SeriesBase クラス

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**
派生クラス **Series**, **BoxWhisker**, **TrendLineBase**, **Waterfall**, **FlexRadarSeries**, **FinancialSeries**, **Fibonacci**, **FibonacciArcs**, **FibonacciFans**, **FibonacciTimeZones**, **OverlayIndicatorBase**

チャートに表示するデータポイントの系列を表します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): SeriesBase
```

SeriesBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **SeriesBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

型 **any**

axisX

系列のx軸を取得または設定します。

型 **Axis**

axisY

系列のy軸を取得または設定します。

型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

型 **string**

- hostElement

系列のホスト要素を取得します。

型 **SVGGElement**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

型 **any**

- legendElement

系列の凡例要素を取得します。

型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

型 **string**

- style

系列のスタイルを取得または設定します。

型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

型 **`Marker`**

symbolSize

この**`Series`** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

型 **`SeriesVisibility`**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

`Point` を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**`Point`** 。

戻り値 **`Point`**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

引数 **EventArgs**

SeriesEventArgs クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

Series イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- empty
- series

コンストラクタ

constructor

```
constructor(series: SeriesBase): SeriesEventArgs
```

SeriesEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **series**: **SeriesBase**
  このイベントの影響を受ける**Series** オブジェクトを指定します。

戻り値 **SeriesEventArgs**

プロパティ

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

series

このイベントの影響を受ける**Series** オブジェクトを取得します。

型 **SeriesBase**

SeriesRenderingEventArgs クラス

ファイル wijmo.chart.js
モジュール **wijmo.chart**
基本クラス **RenderEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

Series イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- count
- empty
- engine
- index

コンストラクタ

constructor

```
constructor(engine: IRenderEngine, index: number, count: number): SeriesRenderingEventArgs
```

SeriesRenderingEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **engine**: **IRenderEngine**
  （**IRenderEngine**）使用するレンダリングエンジン。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

戻り値 **SeriesRenderingEventArgs**

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

- count

レンダリングする系列の合計数を取得します。

型 **number**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

engine

チャート要素のレンダリングに使用する**IRenderEngine** オブジェクトを取得します。

継承元 **RenderEventArgs**
型 **IRenderEngine**

index

レンダリングする系列のインデックスを取得します。

型 **number**

IRenderEngine インターフェイス

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

基本描画ルーチンを実行するレンダリングエンジンを表します。

プロパティ

- element
- fill
- fontFamily
- fontSize
- stroke
- strokeWidth
- textFill

メソッド

- addClipRect
- beginRender
- drawDonutSegment
- drawEllipse
- drawImage
- drawLine
- drawLines
- drawPieSegment
- drawPolygon
- drawRect
- drawSplines
- drawString
- drawStringRotated
- endGroup
- endRender
- measureString
- setViewportSize
- startGroup

プロパティ

element

レンダリングされた要素を取得します。

型 **Element**

fill

要素の塗りつぶしに使用される色を取得または設定します。

型 **string**

- fontFamily

テキスト出力のフォントファミリを取得または設定します。

型 **string**

- fontSize

テキスト出力のフォントサイズを取得または設定します。

型 **string**

- stroke

要素の輪郭の描画に使用される色を取得または設定します。

型 **string**

- strokeWidth

輪郭の太さを取得または設定します。

型 **number**

- textFill

テキストの色を取得または設定します。

型 **string**

メソッド

- addClipRect

```
addClipRect(clipRect: Rect, id: string): void
```

コンテキストにクリッピング四角形を追加します。

パラメーター

- **clipRect**: **Rect**
  クリッピング四角形を表します。
- **id**: **string**
  クリッピング四角形のID。

戻り値 **void**

- beginRender

```
beginRender(): void
```

ビューポートをクリアしてレンダリングサイクルを開始します。

戻り値 **void**

drawDonutSegment

```
drawDonutSegment(cx: number, cy: number, radius: number, innerRadius: number, startAngle: number, sweepAngle: number, className?: strin
g, style?: any, clipPath?: string): void
```

ドーナツ円グラフセグメントを描画します。

パラメーター

- **cx**: **number**
  セグメントの中心の X 座標。
- **cy**: **number**
  セグメントの中心の Y 座標。
- **radius**: **number**
  セグメントの外側半径。
- **innerRadius**: **number**
  セグメントの内側半径。
- **startAngle**: **number**
  セグメントの開始角度（度単位）。
- **sweepAngle**: **number**
  セグメントのスイープ角度（度単位、時計回り）。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。
- **clipPath**: **string** OPTIONAL
  クリッピングパスとして使用するパスのId。

戻り値 **void**

drawEllipse

```
drawEllipse(cx: number, cy: number, rx: number, ry: number, className?: string, style?: any): void
```

楕円を描画します。

パラメーター

- **cx**: **number**
  楕円の中心の x 座標。
- **cy**: **number**
  楕円の中心の y 座標。
- **rx**: **number**
  x 半径。
- **ry**: **number**
  y 半径。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。

戻り値 **void**

drawImage

```
drawImage(href: string, x: number, y: number, w: number, h: number): void
```

画像を描画します。

パラメーター

- **href**: **string**
  描画する画像のURL。
- **x**: **number**
  画像の境界四角形の左座標。
- **y**: **number**
  画像の境界四角形の下座標。
- **w**: **number**
  画像の幅。
- **h**: **number**
  画像の高さ。

戻り値 **void**

drawLine

```
drawLine(x1: number, y1: number, x2: number, y2: number, className?: string, style?: any): void
```

線を描画します。

パラメーター

- **x1**: **number**
  最初の点の x 座標。
- **y1**: **number**
  最初の点の y 座標。
- **x2**: **number**
  2 番目の点の x 座標。
- **y2**: **number**
  2 番目の点の y 座標。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。

戻り値 **void**

drawLines

```
drawLines(xs: number[], ys: number[], className?: string, style?: any, clipPath?: string): void
```

一連の線分を描画します。

パラメーター

- **xs**: **number[]**
  X座標の配列。
- **ys**: **number[]**
  Y座標の配列。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。
- **clipPath**: **string** OPTIONAL
  クリッピングパスとして使用するパスのId。

戻り値 **void**

drawPieSegment

```
drawPieSegment(cx: number, cy: number, radius: number, startAngle: number, sweepAngle: number, className?: string, style?: any, clipPath?: string): void
```

円グラフセグメントを描画します。

パラメーター

- **cx**: **number**
  セグメントの中心の X 座標。
- **cy**: **number**
  セグメントの中心の Y 座標。
- **radius**: **number**
  セグメントの半径。
- **startAngle**: **number**
  セグメントの開始角度（度単位）。
- **sweepAngle**: **number**
  セグメントのスイープ角度（度単位、時計回り）。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。
- **clipPath**: **string** OPTIONAL
  クリッピングパスとして使用するパスのId。

戻り値 **void**

drawPolygon

```
drawPolygon(xs: number[], ys: number[], className?: string, style?: any, clipPath?: string): void
```

多角形を描画します。

パラメーター

- **xs**: **number[]**
  X座標の配列。
- **ys**: **number[]**
  Y座標の配列。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。
- **clipPath**: **string** OPTIONAL
  クリッピングパスとして使用するパスのId。

戻り値 **void**

drawRect

```
drawRect(x: number, y: number, w: number, h: number, className?: string, style?: any, clipPath?: string): void
```

四角形を描画します。

パラメーター

- **x**: **number**
  描画する四角形の左部。
- **y**: **number**
  描画する四角形の底部。
- **w**: **number**
  四角形の幅。
- **h**: **number**
  四角形の高さ。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。
- **clipPath**: **string** OPTIONAL
  クリッピングパスとして使用するパスのId。

戻り値 **void**

drawSplines

```
drawSplines(xs: number[], ys: number[], className?: string, style?: any, clipPath?: string): void
```

一連のスプライン分（滑らかなパス）を描画します。

パラメーター

- **xs**: **number[]**
  X座標の配列。
- **ys**: **number[]**
  Y座標の配列。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。
- **clipPath**: **string** OPTIONAL
  クリッピングパスとして使用するパスのId。

戻り値 **void**

drawString

```
drawString(s: string, pt: Point, className?: string, style?: any): void
```

文字列を描画します。

パラメーター

- **s**: **string**
  描画する文字列。
- **pt**: **Point**
  文字列の基準点。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。

戻り値 **void**

drawStringRotated

```
drawStringRotated(s: string, pt: Point, center: Point, angle: number, className?: string, style?: any): void
```

回転した文字列を描画します。

パラメーター

- **s**: **string**
  描画する文字列。
- **pt**: **Point**
  文字列を描画する用の基準点。
- **center**: **Point**
  文字列を回転する用の基準点。
- **angle**: **number**
  回転角度（度単位、時計回り）。
- **className**: **string** OPTIONAL
  要素に適用されるクラス名。
- **style**: **any** OPTIONAL
  要素に適用されるスタイルオブジェクト。

戻り値 **void**

endGroup

```
endGroup(): void
```

グループを終了します。

戻り値 **void**

endRender

```
endRender(): void
```

レンダリングサイクルを完了します。

戻り値 **void**

measureString

```
measureString(s: string, className?: string, groupName?: string, style?: any): Size
```

文字列を測定します。

パラメーター

- **s**: **string**
  測定する文字列。
- **className**: **string** OPTIONAL
  文字列を測定するときに使用するクラス名。
- **groupName**: **string** OPTIONAL
  文字列を測定するときに使用するグループ名。
- **style**: **any** OPTIONAL
  文字列を測定するときに使用するスタイルオブジェクト。

戻り値 **Size**

setViewportSize

```
setViewportSize(w: number, h: number): void
```

ビューポートのサイズを設定します。

パラメーター

- **w**: **number**
  ビューポートの幅。
- **h**: **number**
  ビューポートの高さ。

戻り値 **void**

startGroup

```
startGroup(className?: string, clipPath?: string, createTransform?: boolean): void
```

グループを開始します。

パラメーター

- **className**: **string** OPTIONAL
  新しいグループに適用するクラス名。
- **clipPath**: **string** OPTIONAL
  クリッピングパスとして使用するパスのId。
- **createTransform**: **boolean** OPTIONAL
  グループの新しい変換を作成するかどうか。

戻り値 **void**

AxisType 列挙体

ファイル wijmo.chart.js
モジュール **wijmo.chart**

軸タイプを指定します。

メンバー

名前	値	説明
X	0	カテゴリー軸（通常は横軸）。
Y	1	値軸（通常は縦軸）。

ChartElement 列挙体

ファイル wijmo.chart.js
モジュール **wijmo.chart**

hitTestメソッドで検出するチャート要素のタイプを指定します。

メンバー

名前	値	説明
PlotArea	0	軸の内側の領域。
AxisX	1	X軸。
AxisY	2	Y軸。
ChartArea	3	コントロールの内部で軸の外部の領域。
Legend	4	チャートの凡例。
Header	5	チャートのヘッダ。
Footer	6	チャートのフッタ。
Series	7	チャートの系列。
SeriesSymbol	8	チャートの系列シンボル。
DataLabel	9	データラベル。
None	10	チャート要素なし。

ChartType 列挙体

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

チャートタイプを指定します。

メンバー

名前	値	説明
Column	0	縦棒を表示して、カテゴリ間で項目の値を比較できるようにします。
Bar	1	横棒を表示します。
Scatter	2	X座標とY座標を使用して、データに含まれるパターンを表示します。
Line	3	一定期間のトレンドまたはカテゴリ間のトレンドを表示します。
LineSymbols	4	各データポイントにシンボルを使用する折れ線グラフを表示します。
Area	5	線の下の領域が色で塗りつぶされた折れ線グラフを表示します。
Bubble	6	シンボルのサイズが3番目のデータ値によって決定される散布図 を表示します。 このチャートタイプのデータは、**`FlexChart`** プロパティまたは**`Series`** **binding**プロパティを使用して、 "yProperty, bubbleSizeProperty"形式のカンマ区切り値で定義できます。
Candlestick	7	高値、安値、始値、終値を持つ項目を表示します。 ヒゲ線のサイズは高値と安値で決定され、 棒のサイズは、始値と終値で決定されます。 棒は、終値が始値より高いか安いかに応じて、 異なる色で表示されます。 このチャートタイプのデータは、**`FlexChart`** プロパティまたは**`Series`** **binding**プロパティを使用して、 "highProperty, lowProperty, openProperty, closeProperty"形式のカンマ区切り値で定義できます。
HighLowOpenClose	8	ローソク足チャートと同じ情報を表示しますが、 始値が左向きの線、終値が右向きの線 で表されます。 このチャートタイプのデータは、 **`FlexChart`** プロパティまたは**`Series`** **binding**プロパティを使用して、 "yProperty, bubbleSizeProperty"形式のカンマ区切り値で定義できます。
Spline	9	折れ線ではなく曲線でデータポイントをプロットする折れ線グラフです。
SplineSymbols	10	各データポイントにシンボルを使用するスプライングラフを表示します。
SplineArea	11	線の下の領域が色で塗りつぶされたスプライングラフを表示します。
Funnel	12	通常、販売パイプラインなどのように過程の段階を表すファンネルチャートを表示します。

ImageFormat 列挙体

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

埋め込まれたBase64エンコードバイナリデータを含む画像の形式を指定します。

メンバー

名前	値	説明
Png	0	W3C Portable Network Graphics（PNG）画像形式を取得します。
Jpeg	1	Joint Photographic Experts Group（JPEG）画像形式を取得します。
Svg	2	Scalable Vector Graphics（SVG）画像形式を取得します。

LabelPosition 列挙体

ファイル wijmo.chart.js
モジュール **wijmo.chart**

データラベルのチャート上での位置を指定します。

メンバー

名前	値	説明
None	0	データラベルは表示されません。
Left	1	データポイントの左に表示されます。
Top	2	データポイントの上に表示されます。
Right	3	データポイントの右に表示されます。
Bottom	4	データポイントの下に表示されます。
Center	5	データポイントを中心として表示されます。

LineMarkerAlignment 列挙体

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

LineMarker の配置を指定します。

メンバー

名前	値	説明
Auto	2	LineMarkerがプロット領域の境界内にとどまるように配置を自動的に調整します。
Right	0	LineMarkerをポインタの右に配置します。
Left	1	LineMarkerをポインタの左に配置します。
Bottom	4	LineMarkerをポインタの下に配置します。
Top	6	LineMarkerをポインタの上に配置します。

LineMarkerInteraction 列挙体

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

LineMarker がユーザー操作とどのように連動するかを指定します。

メンバー

名前	値	説明
None	0	連動なし。ユーザーはクリックによって位置を指定します。
Move	1	**`LineMarker`** はポインタとともに移動します。
Drag	2	ユーザーが線をドラッグすると**`LineMarker`** が移動します。

LineMarkerLines 列挙体

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

LineMarker によって表示される線の方向を指定します。

メンバー

名前	値	説明
None	0	線を表示しません。
Vertical	1	垂直線を表示します。
Horizontal	2	水平線を表示します。
Both	3	垂直線と水平線の両方を表示します。

Marker 列挙体

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

symbolMarker プロパティに使用するマーカーのタイプを指定します。

Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

メンバー

名前	値	説明
Dot	0	円を使用して各データポイントをマークします。
Box	1	正方形を使用して各データポイントをマークします。

OverlappingLabels 列挙体

ファイル wijmo.chart.js
モジュール **wijmo.chart**

重なっているラベルの処理方法を指定します。

メンバー

名前	値	説明
Auto	0	重なっているラベルを非表示にします。
Show	1	重なっているラベルも含めて、すべてのラベルを表示します。

PieLabelPosition 列挙体

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

データラベルの円グラフ上での位置を指定します。

メンバー

名前	値	説明
None	0	データラベルなし。
Inside	1	パイスライスの内側に表示されます。
Center	2	パイスライスの中央に表示されます。
Outside	3	パイスライスの外側に表示されます。
Radial	4	項目を円グラフセグメント内でセグメントの角度に応じた方向に表示します。
Circular	5	項目を円グラフセグメント内で円周方向に表示します。

Position 列挙体

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

軸または凡例のチャート上での位置を指定します。

メンバー

名前	値	説明
None	0	項目が表示されません。
Left	1	チャートの左に表示されます。
Top	2	チャートの上に表示されます。
Right	3	チャートの右に表示されます。
Bottom	4	チャートの下に表示されます。
Auto	5	項目は自動的に配置されます。

SelectionMode 列挙体

ファイル wijmo.chart.js
モジュール **wijmo.chart**

ユーザーがチャートをクリックしたときに何が選択されるかを指定します。

メンバー

名前	値	説明
None	0	ユーザーがチャートをクリックしたときに系列もデータポイントも選択しません。
Series	1	ユーザーがチャートの系列をクリックしたときに**Series** 全体を選択します。
Point	2	ユーザーがグラフをクリックしたときにデータポイントが選択されます。 折れ線グラフ、面グラフ、スプライングラフ、スプライン面グラフは 個々のデータポイントをレンダリングしないので、これらのチャートタイプでこの設定を行っても 何も選択されません。

SeriesVisibility 列挙体

ファイル wijmo.chart.js
モジュール **wijmo.chart**

系列を表示するかどうか、表示する場合はどこに表示するかを指定します。

メンバー

名前	値	説明
Visible	0	系列をプロットと凡例に表示します。
Plot	1	系列をプロットのみに表示します。
Legend	2	系列を凡例のみに表示します。
Hidden	3	系列を表示しません。

Stacking 列挙体

ファイル wijmo.chart.js
モジュール **wijmo.chart**

チャートのデータ値を積み重ねるかどうか、積み重ねる場合はどのように積み重ねるかを指定します。

メンバー

名前	値	説明
None	0	積み重ねなし。 各系列オブジェクトは個別にプロットされます。
Stacked	1	積み重ねグラフは、それぞれの値が合計にどの程度寄与しているかを示します。
Stacked100pc	2	各値が全体にどの程度影響するかを示す100%積層グラフにします。 全体に対する各系列の割合を相対サイズで表します。

TickMark 列挙体

ファイル `wijmo.chart.js`
モジュール **wijmo.chart**

軸の目盛りマークを表示するかどうかと、表示する場合はその場所を指定します。

メンバー

名前	値	説明
None	0	目盛を表示しません。
Outside	1	目盛をプロット領域の外側に表示します
Inside	2	目盛をプロット領域の内側に表示します。
Cross	3	目盛は軸と交差します。

wijmo.chart.analytics モジュール

ファイル `wijmo.chart.analytics.js`
モジュール **wijmo.chart.analytics**

チャートに**TrendLine** 、**MovingAverage** 、**FunctionSeries** などの分析機能を追加するクラスを定義します。

クラス

- BoxWhisker
- ErrorBar
- FunctionSeries
- MovingAverage
- ParametricFunctionSeries
- TrendLine
- TrendLineBase
- Waterfall
- YFunctionSeries

列挙体

- ErrorAmount
- ErrorBarDirection
- ErrorBarEndStyle
- MovingAverageType
- QuartileCalculation
- TrendLineFitType

BoxWhisker クラス

ファイル wijmo.chart.analytics.js
モジュール **wijmo.chart.analytics**
基本クラス **SeriesBase**
派生クラス **WjFlexChartBoxWhisker**
表示 ☐継承されたメンバー ☐イベント発生元

箱ひげ図系列を表します。

通常、**BoxWhisker** 系列は、複数の数値データセット間の分布を比較するために使用されます。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- gapWidth
- groupWidth
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- meanLineStyle
- meanMarkerStyle
- name
- quartileCalculation
- showInnerPoints
- showMeanLine
- showMeanMarker
- showOutliers
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize

- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): BoxWhisker
```

BoxWhisker クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **BoxWhisker**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- gapWidth

グループ間のギャップ幅をパーセント値で指定する値を取得または設定します。

このプロパティのデフォルト値は0.1です。最小値は0で、最大値は1です。

型 **number**

- groupWidth

グループ幅をパーセント値で指定する値を取得または設定します。

このプロパティのデフォルト値は0.8です。最小値は0で、最大値は1です。

型 **number**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- meanLineStyle

平均線のスタイルを指定する値を取得または設定します。

型 **any**

- meanMarkerStyle

平均マーカーのスタイルを指定する値を取得または設定します。

型 **any**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- quartileCalculation

四分位数計算方法を指定する値を取得または設定します。

型 **QuartileCalculation**

showInnerPoints

系列の各ポイントの内側データポイントを表示するかどうかを決定する値を取得または設定します。

型 **boolean**

showMeanLine

平均線を表示するかどうかを指定する値を取得または設定します。

型 **boolean**

showMeanMarker

平均マーカーを表示するかどうかを指定する値を取得または設定します。

型 **boolean**

showOutliers

異常値を表示するかどうかを指定する値を取得または設定します。

異常値は、第1と第3の四分位間の範囲外にある内側ポイントです。

型 **boolean**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

ErrorBar クラス

ファイル	wijmo.chart.analytics.js
モジュール	**wijmo.chart.analytics**
基本クラス	**Series**
派生クラス	**WjFlexChartErrorBar**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart の**ErrorBar** 系列を表します。

ErrorBar 系列を使用すると、誤差の範囲や標準偏差を一目で確認することができます。これらは、標準誤差量、パーセント値、または標準偏差として表示することができます。

必要に応じて、誤差値を明示的に設定して、正確な量を表示することもできます。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- direction
- endStyle
- errorAmount
- errorBarStyle
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- value
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): ErrorBar
```

ErrorBar クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **ErrorBar**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

chartType

特定の系列のチャートタイプを取得または設定します。これはチャート全体に設定されたチャートタイプをオーバーライドします。 The default value for this property is **null**, which causes the series to use chart type defined by the parent chart.

継承元 **Series**
型 **ChartType**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

direction

誤差範囲バーの方向を指定する値を取得または設定します。

型 **ErrorBarDirection**

endStyle

誤差範囲バーの終点スタイルを指定する値を取得または設定します。

型 **ErrorBarEndStyle**

errorAmount

value プロパティの意味を指定する値を取得または設定します。

型 **ErrorAmount**

errorBarStyle

誤差範囲バーのレンダリングに使用されるスタイルを取得または設定します。

型 **any**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

このSeries をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 SeriesBase
型 number

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 SeriesBase
型 any

value

系列の誤差値を指定する値を取得または設定します。

このプロパティは、errorAmount プロパティと組み合わせて使用されます。

型 any

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 SeriesBase
型 SeriesVisibility

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標のPoint 。

継承元 SeriesBase
戻り値 Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 `SeriesBase`
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 `SeriesBase`
戻り値 `Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

FunctionSeries クラス

ファイル	`wijmo.chart.analytics.js`
モジュール	**wijmo.chart.analytics**
基本クラス	**TrendLineBase**
派生クラス	**ParametricFunctionSeries**, **YFunctionSeries**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart の関数シリーズの基本クラスを表します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- max
- min
- name
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): FunctionSeries
```

FunctionSeries クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **FunctionSeries**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- max

関数を計算するためのパラメータの最大値を取得または設定します。

型 **number**

- min

関数を計算するためのパラメータの最小値を取得または設定します。

型 **number**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

継承元 **TrendLineBase**
型 **number**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

approximate

```
approximate(x: number): number
```

指定されたx値から近似y値を取得します。

パラメーター

- **x**: **number**
  y値の計算に使用されるx値。

継承元 **TrendLineBase**
戻り値 **number**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

MovingAverage クラス

ファイル	wijmo.chart.analytics.js
モジュール	**wijmo.chart.analytics**
基本クラス	**TrendLineBase**
派生クラス	**WjFlexChartMovingAverage**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart および**FinancialChart** の移動平均傾向線を表します。

これは、データセット全体のさまざまなサブセットから一連の平均値を求めることで データポイントを分析する計算です。MovingAverage自体の**type** プロパティを設定して、 **MovingAverage** オブジェクトごとに異なるタイプ定義できます。

MovingAverage クラスには、平均値の計算に使用する期間の数を設定するための **period** プロパティがあります。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- period
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- type
- visibility

メソッド

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): MovingAverage
```

MovingAverage クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **MovingAverage**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

移動平均系列の期間を取得または設定します。 これは、1より大きな整数値に設定する必要があります。

型 **number**

- sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

継承元 **TrendLineBase**
型 **number**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

type

移動平均系列のタイプを取得または設定します。

型 **MovingAverageType**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

approximate

```
approximate(x: number): number
```

指定されたx値から近似y値を取得します。

パラメーター

- **x**: **number**
  y値の計算に使用されるx値。

継承元 **TrendLineBase**
戻り値 **number**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

ParametricFunctionSeries クラス

ファイル	`wijmo.chart.analytics.js`
モジュール	**wijmo.chart.analytics**
基本クラス	**FunctionSeries**
派生クラス	**WjFlexChartParametricFunctionSeries**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart のパラメータ関数シリーズを表します。

ParametricFunctionSeries は、式x=f(t)およびy=f(t)によって定義される関数をプロットします。

x値とy値は、**xFunc** および**yFunc** プロパティに割り当てられた関数によって計算されます。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- max
- min
- name
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- xFunc
- yFunc

メソッド

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): ParametricFunctionSeries
```

ParametricFunctionSeries クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **ParametricFunctionSeries**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- max

関数を計算するためのパラメータの最大値を取得または設定します。

継承元 **FunctionSeries**
型 **number**

- min

関数を計算するためのパラメータの最小値を取得または設定します。

継承元 **FunctionSeries**
型 **number**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

継承元 **TrendLineBase**
型 **number**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

xFunc

x値の計算に使用される関数を取得または設定します。

型 **Function**

yFunc

y値の計算に使用される関数を取得または設定します。

型 **Function**

メソッド

approximate

```
approximate(value: number): void
```

指定された値から近似xおよびyを取得します。

パラメーター

- **value**: **number**
  計算する値。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

TrendLine クラス

ファイル	`wijmo.chart.analytics.js`
モジュール	**wijmo.chart.analytics**
基本クラス	**TrendLineBase**
派生クラス	**WjFlexChartTrendLine**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart または**FinancialChart** の傾向線系列を表します。

傾向線は、データの全体的な傾向をチャートに重ねてわかりやすく示すための線です。

fitTypeプロパティを設定して、**FlexChart** での**TrendLine** 系列ごとに異なるフィッティングタイプを定義できます。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- coefficients
- collectionView
- cssClass
- fitType
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- order
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getEquation
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): TrendLine
```

TrendLine クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **TrendLine**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

coefficients

計算式の係数を取得します。

型 **number**[]

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

fitType

TrendLine のフィッティングタイプを取得または設定します。

型 **TrendLineFitType**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- order

多項式またはフーリエ方程式の項の数を取得または設定します。

この値は、1より大きい整数に設定してください。 これは、fitTypeがwijmo.chart.analytics.TrendLineFitType.Polynomialまたは wijmo.chart.analytics.TrendLineFitType.Fourier に設定されたときに適用されます。

型 **number**

- sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

継承元 **TrendLineBase**
型 **number**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

このSeries をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 SeriesBase
型 number

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 SeriesBase
型 any

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 SeriesBase
型 SeriesVisibility

メソッド

approximate

```
approximate(x: number): number
```

指定されたx値から近似y値を取得します。

パラメーター

- **x**: **number**
  y値の計算に使用されるx値。

戻り値 number

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 SeriesBase
戻り値 Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 `SeriesBase`
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 `SeriesBase`
戻り値 `Rect`

getEquation

```
getEquation(fmt?: Function): void
```

係数の書式設定された式文字列を取得します。

パラメーター

- **fmt**: **Function** OPTIONAL
  係数を文字列に変換するために使用される書式設定関数。 このパラメータはオプションです。

戻り値 **void**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

TrendLineBase クラス

ファイル	`wijmo.chart.analytics.js`
モジュール	**wijmo.chart.analytics**
基本クラス	**SeriesBase**
派生クラス	**FunctionSeries**, **MovingAverage**, **TrendLine**
表示	☐ 継承されたメンバー ☐ イベント発生元

さまざまな傾向線の基本クラスを表します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): TrendLineBase
```

TrendLineBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **TrendLineBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

approximate

```
approximate(x: number): number
```

指定されたx値から近似y値を取得します。

パラメーター

- **x**: **number**
  y値の計算に使用されるx値。

戻り値 **number**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

Waterfall クラス

ファイル wijmo.chart.analytics.js
モジュール **wijmo.chart.analytics**
基本クラス **SeriesBase**
派生クラス **WjFlexChartWaterfall**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart のウォータフォール系列を表します。

通常、**WaterfallSeries** は、開始位置からの増加または減少の様子を一連の 変化量によって表示するために使用されます。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- connectorLines
- cssClass
- hostElement
- intermediateTotalLabels
- intermediateTotalPositions
- interpolateNulls
- itemsSource
- legendElement
- name
- relativeData
- showIntermediateTotal
- showTotal
- start
- startLabel
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- totalLabel
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest

- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Waterfall
```

Waterfall クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Waterfall**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- connectorLines

接続線を表示するかどうかを決定する値を取得または設定します。

型 **boolean**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- intermediateTotalLabels

小計バーのラベルを含むプロパティの名前を取得または設定します。この名前は配列または文字列である必要があります。

このプロパティは、**showIntermediateTotal** プロパティおよび**intermediateTotalPositions** プロパティと組み合わせて使用されます。

型 **any**

- intermediateTotalPositions

小計バーの位置のインデックスを含むプロパティの値を取得または設定します。

このプロパティは、**showIntermediateTotal** プロパティおよび**intermediateTotalLabels** プロパティと組み合わせて使用されます。

型 **number[]**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- relativeData

指定されたデータが相対値（差異）であるかどうかを決定する値を取得または設定します。

型 **boolean**

- showIntermediateTotal

小計バーを表示するかどうかを決定する値を取得または設定します。

このプロパティは、**intermediateTotalPositions** プロパティおよび**intermediateTotalLabels** プロパティと組み合わせて使用されます。

型 **boolean**

- showTotal

チャートの末尾に合計バーを表示するかどうかを決定する値を取得または設定します。

型 **boolean**

- start

開始バーの値を決定する値を取得または設定します。 開始値がnullの場合、開始バーは表示されません。

型 **number**

- startLabel

開始バーのラベルを取得または設定します。

型 **string**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- styles

ウォータフォールのスタイルを取得または設定します。

以下のスタイルがサポートされています。

1. **start**: 開始バーのスタイルを指定します。
2. **total**: 合計バーのスタイルを指定します。
3. **intermediateTotal**: 小計バーのスタイルを指定します。
4. **falling**: 減少バーのスタイルを指定します。
5. **rising**: 増加バーのスタイルを指定します。
6. **connectorLines**: 接続線のスタイルを指定します。

```
waterfall.styles = {
  start: { fill: 'blue', stroke: 'blue' },
  total: { fill: 'yellow', stroke: 'yellow' },
  falling: { fill: 'red', stroke: 'red' },
  rising: { fill: 'green', stroke: 'green' },
  connectorLines: { stroke: 'blue', 'stroke-dasharray': '10, 10' }
}
```

型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

このSeries をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 SeriesBase
型 number

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 SeriesBase
型 any

totalLabel

合計バーのラベルを取得または設定します。

型 string

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 SeriesBase
型 SeriesVisibility

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標のPoint 。

継承元 SeriesBase
戻り値 Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

YFunctionSeries クラス

ファイル	wijmo.chart.analytics.js
モジュール	**wijmo.chart.analytics**
基本クラス	**FunctionSeries**
派生クラス	**WjFlexChartYFunctionSeries**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart のY関数シリーズを表します。

YFunctionSeries は、y=f(x)タイプの式で定義された関数をプロットします。これらの関数は、**func** プロパティを使用して指定されます。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- func
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- max
- min
- name
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): YFunctionSeries
```

YFunctionSeries クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **YFunctionSeries**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- func

Y値の計算に使用される関数を取得または設定します。

型 **Function**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- max

関数を計算するためのパラメータの最大値を取得または設定します。

継承元 **FunctionSeries**
型 **number**

- min

関数を計算するためのパラメータの最小値を取得または設定します。

継承元 **FunctionSeries**
型 **number**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

継承元 **TrendLineBase**
型 **number**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

このSeries をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 SeriesBase
型 number

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 SeriesBase
型 any

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 SeriesBase
型 SeriesVisibility

メソッド

approximate

```
approximate(x: number): number
```

指定されたx値から近似y値を取得します。

パラメーター

- **x**: **number**
  y値の計算に使用されるx値。

戻り値 number

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 SeriesBase
戻り値 Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

ErrorAmount 列挙体

ファイル wijmo.chart.analytics.js
モジュール **wijmo.chart.analytics**

ErrorBar の**ErrorBar** プロパティの意味を指定します。

メンバー

名前	値	説明
FixedValue	0	valueプロパティは誤差を絶対値として表します。
Percentage	1	valueプロパティは誤差をパーセント値として表します。
StandardDeviation	2	valueプロパティは誤差を標準偏差として表します。
StandardError	3	誤差は平均値の標準誤差です（valueプロパティは使用されません）。
Custom	4	誤差値は、**`binding`** プロパティでバインドされるか、'plus'と'minus'値でオブジェクトに設定されます。

ErrorBarDirection 列挙体

ファイル `wijmo.chart.analytics.js`
モジュール **wijmo.chart.analytics**

誤差範囲バーの方向を指定します。

メンバー

名前	値	説明
Both	0	誤差を両方向に表示します。
Minus	1	誤差を負方向のみに表示します。
Plus	2	誤差を正方向のみに表示します。

ErrorBarEndStyle 列挙体

ファイル `wijmo.chart.analytics.js`
モジュール **wijmo.chart.analytics**

誤差範囲バーの終点スタイルを指定します。

メンバー

名前	値	説明
Cap	0	誤差範囲バーの終点にキャップがあります。
NoCap	1	誤差範囲バーは単純な線です。

MovingAverageType 列挙体

ファイル `wijmo.chart.analytics.js`
モジュール **wijmo.chart.analytics**

MovingAverage系列のタイプを指定します。

メンバー

名前	値	説明
Simple	0	最後のn個の値の平均。
Weighted	1	最後のn個の値の加重平均で、1つ前の値の重みが1ずつ減ります。
Exponential	2	最後のn個の値の加重平均で、1つ前の値の重みが指数関数的に減ります。
Triangular	3	最後のn個の値の加重平均で、結果は2回の平滑化を行った単純移動平均と同じです。

QuartileCalculation 列挙体

ファイル `wijmo.chart.analytics.js`
モジュール **wijmo.chart.analytics**

BoxWhisker 系列の四分位数の計算方法を指定します。

メンバー

名前	値	説明
InclusiveMedian	0	四分位数の計算時に中央値を含めます。
ExclusiveMedian	1	四分位数の計算時に中央値を除外します。

TrendLineFitType 列挙体

ファイル `wijmo.chart.analytics.js`
モジュール **wijmo.chart.analytics**

TrendLine 系列のフィッティングタイプを指定します。

メンバー

名前	値	説明
Linear	0	データに最も近似する直線 Y(x) = a * x + b.
Exponential	1	式Y(x) = a * exp(b*x)への回帰フィッティング。
Logarithmic	2	式Y(x) = a * ln(x) + bへの回帰フィッティング。
Power	3	式Y(x) = a * pow(x, b)への回帰フィッティング。
Fourier	4	式Y(x) = a + b * cos(x) + c * sin(x) + d * cos(2*x) + e * sin(2*x) + ...への回帰フィッティング。
Polynomial	5	式Y(x) = a * x^n + b * x^n-1 + c * x^n-2 + ... + zへの回帰フィッティング。 + z.
MinX	6	Xの最小値。
MinY	7	Yの最小値。
MaxX	8	Xの最大値。
MaxY	9	Yの最大値。
AverageX	10	平均X値。
AverageY	11	平均Y値。

wijmo.chart.annotation モジュール

ファイル `wijmo.chart.annotation.js`
モジュール **wijmo.chart.annotation**

FlexChart および**FinancialChart** の**AnnotationLayer** と さまざまな注釈を定義します。

クラス

- AnnotationBase
- AnnotationLayer
- Circle
- Ellipse
- Image
- Line
- Polygon
- Rectangle
- Shape
- Square
- Text

列挙体

- AnnotationAttachment
- AnnotationPosition

AnnotationBase クラス

ファイル wijmo.chart.annotation.js
モジュール **wijmo.chart.annotation**
派生クラス **Shape**, **Text**

AnnotationLayer の注釈の基本クラスを表します。

コンストラクタ

- constructor

プロパティ

- attachment
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): AnnotationBase
```

AnnotationBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **AnnotationBase**

プロパティ

- attachment

注釈の添付方法を取得または設定します。

型 **AnnotationAttachment**

- isVisible

注釈の表示/非表示を取得または設定します。

型 **boolean**

- name

注釈の名前を取得または設定します。

型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

型 **number**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

型 **AnnotationPosition**

- seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

型 **number**

- style

注釈のスタイルを取得または設定します。

型 **any**

- tooltip

注釈のツールチップを取得または設定します。

型 **string**

メソッド

destroy

```
destroy(): void
```

この注釈を破棄します。

戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

戻り値 **void**

AnnotationLayer クラス

ファイル wijmo.chart.annotation.js
モジュール **wijmo.chart.annotation**
派生クラス **WjFlexChartAnnotationLayer**

FlexChart および**FinancialChart** の注釈レイヤを表します。

AnnotationLayerには、テキスト、線、画像、四角形など、さまざまな注釈要素の コレクションが含まれます。 **AnnotationLayer** を使用するには、注釈を作成し、それをレイヤのitemsプロパティに追加します。

コンストラクタ

- constructor

プロパティ

- items

メソッド

- getItem
- getItems

コンストラクタ

constructor

```
constructor(chart: FlexChartCore, options?): AnnotationLayer
```

AnnotationLayer クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartCore**
  AnnotationLayer の添付先のチャート。
- **options**: OPTIONAL
  AnnotationLayer の初期化データを含むJavaScriptオブジェクト。

戻り値 **AnnotationLayer**

プロパティ

- items

AnnotationLayer 内の注釈要素のコレクションを取得します。

型 **ObservableArray**

メソッド

getItem

```
getItem(name: string): AnnotationBase
```

AnnotationLayer 内の注釈要素を名前に基づいて取得します。

パラメーター

- **name**: **string**
  注釈の名前。

戻り値 **AnnotationBase**

getItems

```
getItems(name: string): Array
```

AnnotationLayer 内の複数の注釈要素を名前に基づいて取得します。

パラメーター

- **name**: **string**
  注釈の名前。

戻り値 **Array**

Circle クラス

ファイル	wijmo.chart.annotation.js
モジュール	**wijmo.chart.annotation**
基本クラス	**Shape**
派生クラス	**WjFlexChartAnnotationCircle**
表示	☐継承されたメンバー ☐イベント発生元

AnnotationLayer の円注釈を表します。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- isVisible
- name
- offset
- point
- pointIndex
- position
- radius
- seriesIndex
- style
- tooltip

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Circle
```

Circle 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Circle**

プロパティ

- attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

radius

Circle 注釈の半径を取得または設定します。

型 **number**

seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

メソッド

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

Ellipse クラス

ファイル wijmo.chart.annotation.js
モジュール **wijmo.chart.annotation**
基本クラス **Shape**
派生クラス **WjFlexChartAnnotationEllipse**
表示 ☐継承されたメンバー ☐イベント発生元

AnnotationLayer の楕円注釈を表します。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- height
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Ellipse
```

Ellipse 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Ellipse**

プロパティ

- attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

- content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

- height

Ellipse 注釈の高さを取得または設定します。

型 **number**

- isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

- name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

width

Ellipse 注釈の幅を取得または設定します。

型 **number**

メソッド

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

Image クラス

ファイル	wijmo.chart.annotation.js
モジュール	**wijmo.chart.annotation**
基本クラス	**Shape**
派生クラス	**WjFlexChartAnnotationImage**
表示	☐継承されたメンバー ☐イベント発生元

AnnotationLayer の画像注釈を表します。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- height
- href
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Image
```

Image 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Image**

プロパティ

- attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

- content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

- height

Image 注釈の高さを取得または設定します。

型 **number**

- href

Image 注釈のhrefを取得または設定します。

型 **string**

- isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

- name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

- seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

- tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

- width

Image 注釈の幅を取得または設定します。

型 **number**

メソッド

- destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

Line クラス

ファイル	wijmo.chart.annotation.js
モジュール	**wijmo.chart.annotation**
基本クラス	**Shape**
派生クラス	**WjFlexChartAnnotationLine**
表示	☐継承されたメンバー ☐イベント発生元

AnnotationLayer の線注釈を表します。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- end
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- start
- style
- tooltip

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Line
```

Line 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Line**

プロパティ

- attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

- content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

- end

Line注釈の終了点を取得または設定します。

型 **DataPoint**

- isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

- name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

- seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- start

Line 注釈の開始点を取得または設定します。

型 **DataPoint**

- style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

- tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

メソッド

- destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元	**AnnotationBase**
戻り値	**void**

Polygon クラス

ファイル wijmo.chart.annotation.js
モジュール **wijmo.chart.annotation**
基本クラス **Shape**
派生クラス **WjFlexChartAnnotationPolygon**
表示 ☐継承されたメンバー ☐イベント発生元

AnnotationLayer の多角形注釈を表します。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- isVisible
- name
- offset
- point
- pointIndex
- points
- position
- seriesIndex
- style
- tooltip

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Polygon
```

Polygon 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Polygon**

プロパティ

- attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

- content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

- isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

- name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- points

Polygon 注釈のポイントのコレクションを取得します。

型 **ObservableArray**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

メソッド

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

Rectangle クラス

ファイル wijmo.chart.annotation.js
モジュール **wijmo.chart.annotation**
基本クラス **Shape**
派生クラス **WjFlexChartAnnotationRectangle**
表示 ☐継承されたメンバー ☐イベント発生元

AnnotationLayer の四角形注釈を表します。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- height
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Rectangle
```

Rectangle 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Rectangle**

プロパティ

attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

height

Rectangle 注釈の高さを取得または設定します。

型 **number**

isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

- seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

- tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

- width

Rectangle 注釈の幅を取得または設定します。

型 **number**

メソッド

- destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元	**AnnotationBase**
戻り値	**void**

Shape クラス

ファイル	wijmo.chart.annotation.js
モジュール	**wijmo.chart.annotation**
基本クラス	**AnnotationBase**
派生クラス	**Circle**, **Ellipse**, **Image**, **Line**, **Polygon**, **Rectangle**, **Square**
表示	☐継承されたメンバー ☐イベント発生元

AnnotationLayer の図形注釈の基本クラスを表します。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Shape
```

Shape 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Shape**

プロパティ

- attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

- content

注釈のテキストを取得または設定します。

型 **string**

- isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

- name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

- seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

メソッド

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

Square クラス

ファイル wijmo.chart.annotation.js
モジュール **wijmo.chart.annotation**
基本クラス **Shape**
派生クラス **WjFlexChartAnnotationSquare**
表示 ☐継承されたメンバー ☐イベント発生元

AnnotationLayer の正方形注釈を表します。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- isVisible
- length
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Square
```

Square 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Square**

プロパティ

- attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

- content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

- isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

- length

Square 注釈の長さを取得または設定します。

型 **number**

- name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

メソッド

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

Text クラス

ファイル	wijmo.chart.annotation.js
モジュール	**wijmo.chart.annotation**
基本クラス	**AnnotationBase**
派生クラス	**WjFlexChartAnnotationText**
表示	☐継承されたメンバー ☐イベント発生元

AnnotationLayer のテキスト注釈を表します。

コンストラクタ

- constructor

プロパティ

- attachment
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- text
- tooltip

メソッド

- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Text
```

Text 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Text**

プロパティ

- attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

- isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

- name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

- seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

text

注釈のテキストを取得または設定します。

型 **string**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

メソッド

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

AnnotationAttachment 列挙体

ファイル `wijmo.chart.annotation.js`
モジュール **wijmo.chart.annotation**

注釈の添付方法を指定します。

メンバー

名前	値	説明
DataIndex	0	注釈ポイントの座標は、データ系列インデックスとデータポイントインデックスで定義されます。
DataCoordinate	1	注釈ポイントは、データ座標で指定されます。
Relative	2	注釈ポイントは、コントロール内の相対位置で指定されます。(0,0)が左上隅、(1,1)が右下隅になります。
Absolute	3	注釈ポイントは、コントロールのピクセル座標で指定されます。

AnnotationPosition 列挙体

ファイル wijmo.chart.annotation.js
モジュール **wijmo.chart.annotation**

注釈の位置を指定します。

メンバー

名前	値	説明
Center	0	注釈がターゲットポイントの中心に表示されます。
Top	1	注釈がターゲットポイントの上に表示されます。
Bottom	2	注釈がターゲットポイントの下に表示されます。
Left	4	注釈がターゲットポイントの左に表示されます。
Right	8	注釈がターゲットポイントの右に表示されます。

wijmo.chart.interaction モジュール

ファイル `wijmo.chart.interaction.js`
モジュール **wijmo.chart.interaction**

チャートにインタラクティブ機能を追加するクラスを定義します。

クラス

- ChartGestures
- RangeSelector

列挙体

- InteractiveAxes
- MouseAction
- Orientation

ChartGestures クラス

ファイル `wijmo.chart.interaction.js`
モジュール **wijmo.chart.interaction**
派生クラス **WjFlexChartGestures**

ChartGestures コントロールを使用すると、ユーザーは指定された**FlexChart** でズームまたはパンできるようになります。

ChartGestures コントロールを使用するには、ズームまたはパンする**FlexChart** コントロールを指定します。

var chartGestures = new wijmo.chart.interaction.ChartGestures(chart);

コンストラクタ

- constructor

プロパティ

- enable
- interactiveAxes
- mouseAction
- posX
- posY
- scaleX
- scaleY

メソッド

- remove
- reset

コンストラクタ

constructor

```
constructor(chart: FlexChartCore, options?): ChartGestures
```

ChartGestures クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartCore**
  ユーザーがズームまたはパンできる**FlexChart**。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **ChartGestures**

プロパティ

- enable

ChartGesturesの有効/無効を取得または設定します。

型 **boolean**

- interactiveAxes

ChartGesturesの操作軸を取得または設定します。

型 **`InteractiveAxes`**

- mouseAction

ChartGesturesのマウス操作を取得または設定します。

型 **`MouseAction`**

- posX

軸Xの初期位置を取得または設定します。Scaleが１より小さい場合、この値は軸の初期位置を表します。そうでない場合、Valueは無効です。Valueには０から１までの値を指定する必要があります。

型 **number**

- posY

軸Yの初期位置を取得または設定します。Scaleが１より小さい場合、この値は軸の初期位置を表します。そうでない場合、Valueは無効です。Valueには０から１までの値を指定する必要があります。

型 **number**

- scaleX

軸Xの初期スケールを取得または設定します。このスケールは０より大きく１以下にする必要があります。 スケールは、最小値から最大値までの範囲のどの部分を表示するかを指定します。スケールが１（デフォルト値）の場合は、軸範囲全体が表示されます。

型 **number**

- scaleY

軸Yの初期スケールを取得または設定します。このスケールは０より大きく１以下にする必要があります。 スケールは、最小値から最大値までの範囲のどの部分を表示するかを指定します。スケールが１（デフォルト値）の場合は、軸範囲全体が表示されます。

型 **number**

メソッド

- remove

```
remove(): void
```

チャートから**`ChartGestures`** コントロールを削除します。

戻り値 **void**

reset

```
reset(): void
```

チャートの軸をリセットします。

戻り値　　**void**

RangeSelector クラス

ファイル wijmo.chart.interaction.js
モジュール **wijmo.chart.interaction**
派生クラス **WjFlexChartRangeSelector**

RangeSelector コントロールには範囲セレクタが表示されます。これを使用して、ユーザーは指定された**FlexChart** に表示するデータ範囲を選択できます。

RangeSelector コントロールを使用するには、選択されたデータ範囲を表示する**FlexChart** コントロールを指定します。

最小または最大値が変更されると、**rangeChanged** イベントが発生します。 次に例を示します。

```
var rangeSelector = new wijmo.chart.interaction.RangeSelector(chart);
 rangeSelector.rangeChanged.addHandler(function () {

    // 関連する更新を実行します
    // たとえば、別のチャートの表示範囲を変更します
    update(rangeSelector.min, rangeSelector.max);
 });
```

コンストラクタ

- constructor

プロパティ

- isVisible
- max
- maxScale
- min
- minScale
- orientation
- seamless

メソッド

- onRangeChanged
- remove

イベント

- rangeChanged

コンストラクタ

constructor

```
constructor(chart: FlexChartCore, options?): RangeSelector
```

RangeSelector クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartCore**
  選択された範囲を表示する**FlexChart**。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **RangeSelector**

プロパティ

isVisible

範囲セレクターの表示/非表示設定を取得または設定します。

型 **boolean**

max

範囲の最大値を取得または設定します。 設定されていない場合、最大値は自動的に計算されます。

型 **number**

maxScale

チャート範囲全体の割合として選択できるデータの最大量を取得または設定します。 このプロパティは、0と1の間の値に設定する必要があります。

型 **number**

min

範囲の最小値を取得または設定します。 設定されていない場合、最小値は自動的に計算されます。

型 **number**

minScale

チャート範囲全体の割合として選択できるデータの最小量を取得または設定します。 このプロパティは、0と1の間の値に設定する必要があります。

型 **number**

orientation

範囲セレクターの向きを取得または設定します。

型 **Orientation**

seamless

最小/最大要素を上下にドラッグして反転できるかどうかを決定する値を取得または設定します。

型 **boolean**

メソッド

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

rangeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

remove

```
remove(): void
```

チャートから**RangeSelector** コントロールを削除します。

戻り値 **void**

イベント

rangeChanged

範囲が変更された後に発生します。

引数 **EventArgs**

InteractiveAxes 列挙体

ファイル `wijmo.chart.interaction.js`
モジュール **wijmo.chart.interaction**

チャートジェスチャの操作軸を指定します。

メンバー

名前	値	説明
X	0	操作軸はX。
Y	1	操作軸はY。
XY	2	操作軸はXとYの両方。

MouseAction 列挙体

ファイル `wijmo.chart.interaction.js`
モジュール **wijmo.chart.interaction**

チャートジェスチャのマウス操作を指定します。

メンバー

名前	値	説明
Zoom	0	マウスでチャートをズームします。
Pan	1	マウスでチャートをパンします。

Orientation 列挙体

ファイル `wijmo.chart.interaction.js`
モジュール **wijmo.chart.interaction**

範囲セレクターの向きを指定します。

メンバー

名前	値	説明
X	0	水平、xデータの範囲。
Y	1	垂直、yデータの範囲。

wijmo.chart.animation モジュール

ファイル `wijmo.chart.animation.js`
モジュール **wijmo.chart.animation**

FlexChart、**FinancialChart**、および**FlexPie** の**ChartAnimation** を定義します。

クラス

- ChartAnimation

列挙体

- AnimationMode
- Easing

ChartAnimation クラス

ファイル wijmo.chart.animation.js
モジュール **wijmo.chart.animation**
派生クラス **WjFlexChartAnimation**

FlexChart、**FinancialChart**、および**FlexPie** のアニメーションを表します。

ChartAnimation は、チャートのロードおよび更新中に組み込みのアニメーションを提供します。 アニメーションは、時間、イージング関数、アニメーションモードなどのプロパティを使用して、ユーザーが設定できます。

コンストラクタ

- constructor

プロパティ

- animationMode
- axisAnimation
- duration
- easing

メソッド

- animate

コンストラクタ

constructor

```
constructor(chart: FlexChartBase, options?: any): ChartAnimation
```

ChartAnimation クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartBase**
  ChartAnimation の添付先のチャート。
- **options**: **any** OPTIONAL
  ChartAnimation の初期化データを含むJavaScriptオブジェクト。

戻り値 **ChartAnimation**

プロパティ

- animationMode

プロットポイントを一度に 1 つずつアニメーションするか、系列ごとにアニメーションするか、一度にすべてをアニメーションするかを取得または設定します。 アニメーション全体は時間内に完了します。

型 **AnimationMode**

- axisAnimation

アニメーションが軸に適用されるかどうかを示す値を取得または設定します。

型 **boolean**

- duration

アニメーション全体の長さ（ミリ秒）を取得または設定します。

型 **number**

- easing

アニメーションに適用されるイージング関数を取得または設定します。

型 **Easing**

メソッド

- animate

```
animate(): void
```

アニメーションを実行します。

戻り値 **void**

AnimationMode 列挙体

ファイル wijmo.chart.animation.js
モジュール **wijmo.chart.animation**

チャートのアニメーションモード（一度に 1 ポイントずつ、系列ごと、または一度にすべて）を指定します。

メンバー

名前	値	説明
All	0	すべてのポイントと系列が一度にアニメーションします。
Point	1	アニメーションがポイントごとに実行されます。 複数の系列が同時にアニメーションします。
Series	2	アニメーションが系列ごとに実行されます。 Allオプションと同様に系列全体が一度にアニメーションしますが、一度に 1 つの系列だけがアニメーションします。

Easing 列挙体

ファイル wijmo.chart.animation.js
モジュール **wijmo.chart.animation**

パラメータの時間的な変化率を指定します。

メンバー

名前	値	説明
Linear	0	単純な線形トゥイーニング。イージングおよびアクセラレーションなし。
Swing	1	スイングイージングのイージング式。
EaseInQuad	2	２次イージングインのイージング式。速度ゼロから加速します。
EaseOutQuad	3	２次イージングアウトのイージング式。速度ゼロに減速します。
EaseInOutQuad	4	２次イージングインアウトのイージング式。途中まで加速してから減速します。
EaseInCubic	5	３次イージングインのイージング式。速度ゼロから加速します。
EaseOutCubic	6	３次イージングアウトのイージング式。速度ゼロに減速します。
EaseInOutCubic	7	３次イージングインアウトのイージング式。途中まで加速してから減速します。
EaseInQuart	8	４次イージングインのイージング式。速度ゼロから加速します。
EaseOutQuart	9	４次イージングアウトのイージング式。速度ゼロに減速します。
EaseInOutQuart	10	４次イージングインアウトのイージング式。途中まで加速してから減速します。
EaseInQuint	11	５次イージングインのイージング式。速度ゼロから加速します。
EaseOutQuint	12	５次イージングアウトのイージング式。速度ゼロに減速します。
EaseInOutQuint	13	５次イージングインアウトのイージング式。途中まで加速してから減速します。
EaseInSine	14	サインイージングインのイージング式。速度ゼロから加速します。
EaseOutSine	15	サインイージングアウトのイージング式。速度ゼロに減速します。
EaseInOutSine	16	サインイージングインアウトのイージング式。途中まで加速してから減速します。
EaseInExpo	17	指数イージングインのイージング式。速度ゼロから加速します。
EaseOutExpo	18	指数イージングアウトのイージング式。速度ゼロに減速します。
EaseInOutExpo	19	指数イージングインアウトのイージング式。途中まで加速してから減速します。
EaseInCirc	20	円イージングインのイージング式。速度ゼロから加速します。
EaseOutCirc	21	円イージングアウトのイージング式。速度ゼロに減速します。
EaseInOutCirc	22	円イージングインアウトのイージング式。途中まで加速してから減速します。
EaseInBack	23	バックイージングインのイージング式。速度ゼロから加速します。
EaseOutBack	24	バックイージングアウトのイージング式。速度ゼロに減速します。
EaseInOutBack	25	バックイージングインアウトのイージング式。途中まで加速してから減速します。
EaseInBounce	26	バウンスイージングインのイージング式。速度ゼロから加速します。
EaseOutBounce	27	バウンスイージングアウトのイージング式。速度ゼロに減速します。
EaseInOutBounce	28	バウンスイージングインアウトのイージング式。途中まで加速してから減速します。
EaseInElastic	29	弾性イージングインのイージング式。速度ゼロから加速します。
EaseOutElastic	30	弾性イージングアウトのイージング式。速度ゼロに減速します。
EaseInOutElastic	31	弾性イージングインアウトのイージング式。途中まで加速してから減速します。

wijmo.chart.render モジュール

ファイル `wijmo.chart.render.js`
モジュール **wijmo.chart.render**

wijmoグラフをレンダリングするためのさまざまなレンダーエンジンを提供します。 IEブラウザでチャートのエクスポートを対応させるには、このモジュールをページに追加します。

wijmo.chart.hierarchical モジュール

ファイル `wijmo.chart.hierarchical.js`
モジュール **wijmo.chart.hierarchical**

Sunburst チャートコントロールとそのコントロールに関連付けられたクラスを定義します。

クラス

- Sunburst
- TreeMap

列挙体

- TreeMapType

Sunburst クラス

ファイル wijmo.chart.hierarchical.js
モジュール **wijmo.chart.hierarchical**
基本クラス **FlexPie**
派生クラス **WjSunburst**
表示 ☐継承されたメンバー ☐イベント発生元

Sunburstチャートコントロール。

コンストラクタ

- constructor

プロパティ

- binding
- bindingName
- childItemsPath
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- innerRadius
- isAnimated
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- offset
- palette
- plotMargin
- reversed
- rightToLeft
- selectedIndex
- selectedItemOffset
- selectedItemPosition
- selectionMode
- startAngle
- tooltip

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose

- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FlexPie
```

FlexPie クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  このコントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavascriptオブジェクト。

継承元 **FlexPie**
戻り値 **FlexPie**

プロパティ

- binding

チャート値を含むプロパティの名前を取得または設定します。

継承元 **FlexPie**
型 **string**

- bindingName

データ項目の名前を含むプロパティの名前を取得または設定します。この名前は配列または文字列である必要があります。

型 **any**

- childItemsPath

階層化データの子項目の生成に使用される1つ以上のプロパティの名前を 取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `'items'` ）。

項目が異なる名前を持つ異なるレベルの子項目の場合は、 このプロパティに、各レベルの子項目を含むプロパティの名前から成る 配列を設定します （たとえば、 `[ 'accounts', 'checks', 'earnings' ]` ）。

型 **any**

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

- dataLabel

ポイントのデータラベルを取得または設定します。

継承元 **FlexPie**
型 **PieDataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 `FlexChartBase`
型 **string**

headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 `FlexChartBase`
型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

innerRadius

パイの内側半径のサイズを取得または設定します。

内側半径はパイ半径に対する割合として測定されます。

このプロパティのデフォルト値はゼロです（つまり、円グラフになります）。このプロパティをゼロより大きい値に設定すると、円グラフの中央に穴が開きます（これをドーナツグラフと呼びます）。

The default value for this property is **0**

継承元 `FlexPie`
型 **number**

isAnimated

項目が選択されたときにアニメーションを使用するかどうかを示す値を取得または設定します。

`selectedItemPosition` プロパティおよび**`selectionMode`** プロパティも参照してください。

The default value for this property is **false**.

継承元 `FlexPie`
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

offset

スライスのパイ中心からのオフセットを取得または設定します。

オフセットはパイ半径に対する割合として測定されます。

The default value for this property is **0**.

継承元 **FlexPie**
型 **number**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string**[]

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

- reversed

角度を反転（反時計回り）するかどうかを決定する値を取得または設定します。

デフォルト値はfalseです。この場合、角度は時計回りに測定されます。

The default value for this property is **false**.

継承元 **FlexPie**
型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectedIndex

選択されたスライスのインデックスを取得または設定します。

継承元 **FlexPie**
型 **number**

- selectedItemOffset

選択されたスライスのパイ中心からのオフセットを取得または設定します。

オフセットはパイ半径に対する割合として測定されます。

The default value for this property is **0**.

継承元 **FlexPie**
型 **number**

- selectedItemPosition

選択されたスライスの位置を取得または設定します。

このプロパティを'None'以外の値に設定すると、スライスを選択したときに円グラフが回転します。

円グラフをクリックしたときにスライスが選択されるようにするには、 **selectionMode** プロパティを'Point'に設定する必要があります。

The default value for this property is **Position.None**.

継承元 **FlexPie**
型 **Position**

- selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

startAngle

パイスライスの開始角度（度単位）を取得または設定します。

角度は9時の位置から時計回りに測定されます。

The default value for this property is **0**.

継承元 **FlexPie**
型 **number**

tooltip

チャートの**Tooltip** を取得します。

継承元 **FlexPie**
型 **ChartTooltip**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexPie**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**ImageFormat** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **FlexChartBase**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **FlexChartBase**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

TreeMap クラス

ファイル wijmo.chart.hierarchical.js
モジュール **wijmo.chart.hierarchical**
基本クラス **FlexChartBase**
派生クラス **WjTreeMap**
表示 ☐継承されたメンバー ☐イベント発生元

TreeMap コントロールは、階層構造（ツリー構造）のデータを入れ子矩形のセットとして表示します。ツリーの各ブランチは矩形が与えられ、サブブランチを表すより小さい矩形を含みます。 リーフノード矩形は、指定されたデータの大きさに比例する領域を占有します。 多くの場合、リーフノードはデータの別次元を示すために色付けされています。

TreeMap コントロールを使用するには、**itemsSource** プロパティにデータから成る配列を設定し、**binding** と**bindingName** プロパティを使用して、項目値と項目名を含むプロパティを設定します。

コンストラクタ

- constructor

プロパティ

- binding
- bindingName
- childItemsPath
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- maxDepth
- palette
- plotMargin
- rightToLeft
- selectionMode
- tooltip
- type

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate

- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged

コンストラクタ

constructor

```
constructor(element: any, options?): TreeMap
```

TreeMap クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavascriptオブジェクト。

戻り値 **TreeMap**

プロパティ

binding

チャートの値を含むデータ項目のプロパティの名前を取得または設定します。

連結プロパティは、他のノードの値と比較してノードのサイズを計算するために使用されます。 このプロパティには、数値データを設定する必要があります。

型 **string**

bindingName

データ項目の名前を含むプロパティの名前を取得または設定します。 bindingNameプロパティは、ノードの名前を表示するために使用されます。この名前は配列または文字列である必要があります。

型 **any**

childItemsPath

階層化データの子項目の生成に使用される1つ以上のプロパティの名前を 取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `'items'` ）。

項目が異なる名前を持つ異なるレベルの子項目の場合は、 このプロパティに、各レベルの子項目を含むプロパティの名前から成る 配列を設定します （たとえば、 `[ 'accounts', 'checks', 'earnings' ]` ）。

型 **any**

collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

dataLabel

ツリーマップの**DataLabel** を取得または設定します。

型 **DataLabel**

footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

maxDepth

ドロップダウンに表示する項目の最大数を取得または設定します。 これらのレベルが現在の平面に平坦化されます。 ツリーマップのレベルがこの値よりも高い場合、ユーザーは上下に移動する必要があります。

型 **number**

palette

ツリーマップで使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

または、titleColor、maxColor、minColorを別々に含みます。次に例を示します。

```
chart.palette = [{
     titleColor: '#00277d',
     maxColor: 'rgba(0,39,125,0.7)',
     minColor: 'rgba(168,187,230,0.7)'
 }, {
     titleColor: '#7d1f00',
     maxColor: 'rgba(125,21,0,0.7)',
     minColor: 'rgba(230,183,168,0.7)'
 }, {
     titleColor: '#007d27',
     maxColor: 'rgba(0,125,39,0.7)',
     minColor: 'rgba(168,230,188,0.7)'
 }];
```

型 **string[]**

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectionMode

selectionModeは、TreeMapコントロールでは機能しません。

型 **SelectionMode**

- tooltip

チャートの**Tooltip** を取得します。

型 **ChartTooltip**

- type

ツリーマップの**TreeMapType** を取得または設定します。

型 **TreeMapType**

メソッド

- addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**ImageFormat** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **FlexChartBase**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **FlexChartBase**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

TreeMapType 列挙体

ファイル wijmo.chart.hierarchical.js
モジュール **wijmo.chart.hierarchical**

ツリーマップタイプを指定します。

メンバー

名前	値	説明
Squarified	0	正方形のツリーマップを表示します。
Horizontal	1	横長長方形のツリーマップを表示します。
Vertical	2	縦長長方形のツリーマップを表示します。

wijmo.chart.radar モジュール

ファイル `wijmo.chart.radar.js`
モジュール **wijmo.chart.radar**

FlexRadar コントロールとそのコントロールに関連付けられたクラスを定義します。

クラス

- FlexRadar
- FlexRadarAxis
- FlexRadarSeries

列挙体

- RadarChartType

FlexRadar クラス

ファイル wijmo.chart.radar.js
モジュール **wijmo.chart.radar**
基本クラス **FlexChartCore**
派生クラス **WjFlexRadar**
表示 ☐継承されたメンバー ☐イベント発生元

レーダーチャートコントロール。

コンストラクタ

- constructor

プロパティ

- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- interpolateNulls
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- palette
- plotAreas
- plotMargin
- reversed
- rightToLeft
- selection
- selectionMode
- series
- stacking
- startAngle
- symbolSize
- tooltip
- totalAngle

メソッド

- addEventListener

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FlexRadar
```

FlexRadar クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  このコントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **FlexRadar**

プロパティ

axes

Axis オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

axisX

メインのX軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

axisY

メインのY軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

binding

Yの値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

bindingX

Xデータ値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

chartType

作成するレーダーチャートのタイプを取得または設定します。

型 **RadarChartType**

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

- dataLabel

ポイントのデータラベルを取得または設定します。

継承元 **FlexChartCore**
型 **DataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 `FlexChartCore`
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

legendToggle

凡例項目をクリックしたときにチャート上の系列の表示/非表示を切り替えるかどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 **FlexChartCore**
型 **boolean**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string[]**

plotAreas

PlotArea オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **PlotAreaCollection**

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

reversed

角度を反転（反時計回り）するかどうかを決定する値を取得または設定します。

デフォルト値はfalseです。この場合、角度は時計回りに測定されます。

型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selection

選択されているチャート系列を取得または設定します。

継承元 **FlexChartCore**
型 **SeriesBase**

selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

series

Series オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

stacking

系列オブジェクトを積み重ねるかどうかを決定する値と、積み重ねる場合はその方法を取得または設定します。

型 **Stacking**

startAngle

レーダーの開始角度（度単位）を取得または設定します。

角度は、12時の位置から時計回りに測定されます。

型 **number**

symbolSize

このFlexChart のすべてのSeriesオブジェクトに使用されるシンボルのサイズを取得または設定します。

このプロパティは、各Series オブジェクトのsymbolSizeプロパティによってオーバーライドできます。

The default value for this property is **10** pixels.

継承元 FlexChartCore
型 number

tooltip

チャートのTooltip オブジェクトを取得します。

ツールチップの内容は、次のパラメータを含むテンプレートを使用して生成されます。

- **propertyName**： このポイントによって表されるデータオブジェクトの任意のプロパティ。
- **seriesName**： データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**： データポイントのインデックス。
- **value**： データポイントの**値**（**FlexChart** の場合はy値、**FlexPie** の場合は項目値）。
- **x**： データポイントの**x**値（FlexChartのみ）。
- **y**： データポイントの**y**値（FlexChartのみ）。
- **name**： データポイントの**名前**（**FlexChart** の場合はx値、**FlexPie** の場合は凡例エントリ）。

テンプレートを変更するには、ツールチップのコンテンツプロパティに新しい値を割り当てます。 次に例を示します。

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

チャートのツールチップを無効にできます。それには、テンプレートを空の文字列に設定します。

tooltip プロパティを使用して、次のように、**showDelay** や**hideDelay** などの ツールチップパラメータをカスタマイズすることもできます。

```
chart.tooltip.showDelay = 1000;
```

詳細とオプションについては、**ChartTooltip** プロパティを参照してください。

継承元 FlexChartCore
型 ChartTooltip

totalAngle

レーダーの合計角度（度単位）を取得または設定します。デフォルト値は360です この値は、0より大きく、360以下である必要があります。

型 number

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Point をデータ座標からコントロール座標に変換します。

パラメーター

- **pt**: **any**
  データ座標の**Point** 、またはデータ座標のポイントのX座標。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

seriesVisibilityChanged イベントを発生させます。

パラメーター

- **e**: **SeriesEventArgs**
  イベントデータを含む**SeriesEventArgs** オブジェクト。

継承元 **FlexChartCore**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

pointToData

```
pointToData(pt: any, y?: number): Point
```

Point をコントロール座標からチャートデータ座標に変換します。

パラメーター

- **pt**: **any**
  変換するポイント（コントロール座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**`ImageFormat`** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **`FlexChartBase`**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **`FlexChartBase`**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

seriesVisibilityChanged

系列の表示/非表示が変更されたときに発生します。たとえば、legendToggleプロパティをtrueに設定したり、ユーザーが凡例をクリックしたときです。

継承元 **FlexChartCore**
引数 **SeriesEventArgs**

FlexRadarAxis クラス

ファイル	wijmo.chart.radar.js
モジュール	**wijmo.chart.radar**
基本クラス	**Axis**
派生クラス	**WjFlexRadarAxis**
表示	☐継承されたメンバー ☐イベント発生元

レーダーチャート内の軸を表します。

コンストラクタ

- constructor

プロパティ

- actualMax
- actualMin
- axisLine
- axisType
- binding
- format
- hostElement
- itemFormatter
- itemsSource
- labelAlign
- labelAngle
- labelPadding
- labels
- logBase
- majorGrid
- majorTickMarks
- majorUnit
- max
- min
- minorGrid
- minorTickMarks
- minorUnit
- name
- origin
- overlappingLabels
- plotArea
- position
- reversed
- title

メソッド

- convert
- convertBack
- onRangeChanged

イベント

- rangeChanged

コンストラクタ

constructor

```
constructor(position?: Position): Axis
```

Axis クラスの新しいインスタンスを初期化します。

パラメーター

- **position**: **Position** OPTIONAL
  チャート上での軸の位置。

継承元 **Axis**
戻り値 **Axis**

プロパティ

actualMax

実際の軸の最大を取得します。

これは、数値またはDateオブジェクト（時間ベースのデータの場合）を返します。

継承元 **Axis**
型 **any**

actualMin

実際の軸の最小を取得します。

これは、数値またはDateオブジェクト（時間ベースのデータの場合）を返します。

継承元 **Axis**
型 **any**

axisLine

軸線が表示されるかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Axis**
型 **boolean**

axisType

軸タイプを取得します。

継承元 **Axis**
型 **AxisType**

binding

軸ラベルで使用する**itemsSource** プロパティの カンマ区切りのプロパティ名を取得または設定します。

最初の名前は軸の値を指定し、2番目は対応する軸ラベルを表します。デフォルト値は'value,text'です。

継承元 **Axis**
型 **string**

format

軸ラベルに使用される書式文字列を取得または設定します （**Globalize** を参照）。

継承元 **Axis**
型 **string**

hostElement

軸のホスト要素を取得します。

継承元 **Axis**
型 **SVGGElement**

itemFormatter

軸ラベルのitemFormatter関数を取得または設定します。

指定された場合、関数は次の2つのパラメータをとります。

- **render engine**：ラベルの書式設定に使用される**IRenderEngine** オブジェクト。
- **current label**：以下のプロパティを含むオブジェクト。
  - **value**：書式設定する軸ラベルの値。
  - **text**：ラベルに使用するテキスト。
  - **pos**：ラベルがレンダリングされる コントロール座標内の位置。
  - **cls**：ラベルに適用されるCSSクラス。

この関数は、プロパティが変更されるラベルの ラベルパラメータを返します。

次に例を示します。

```
chart.axisY.itemFormatter = function(engine, label) {
    if (label.val > 5){
        engine.textFill = 'red'; // 赤色のテキスト
        label.cls = null; // デフォルトのCSSなし
     }
    return label;
}
```

継承元 **Axis**
型 **Function**

itemsSource

軸ラベルの項目ソースを取得または設定します。

プロパティの名前は、**binding** プロパティによって指定されます。

次に例を示します。

```
// Axis.bindingのデフォルト値は'value,text'です
 chart.axisX.itemsSource = [ { value:1, text:'one' }, { value:2, text:'two' } ];
```

<

継承元 **Axis**
型 **any**

labelAlign

ラベルの配置を取得または設定します。

デフォルトでは、ラベルは中央に配置されます。サポートされている値は、'left'および'right'（x軸の場合）と'top'および'bottom'（y軸の場合）です。

継承元 **Axis**
型 **string**

labelAngle

軸ラベルの回転角度を取得または設定します。

角度は度単位で測定されます。有効な値の範囲は-90〜90です。

継承元 **Axis**
型 **number**

labelPadding

ラベルのパディングを取得または設定します。 The default value for this property is **5** pixels.

継承元 **Axis**
型 **number**

labels

軸ラベルを表示すかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Axis**
型 **boolean**

logBase

軸の対数の底を取得または設定します。

底が指定されていない場合、その軸は線形スケールになります。

logBase プロパティを使用すると、原点の周囲に集まっているデータが広がります。これは一部の金融データセットや経済データセットでよく見られます。

継承元 **Axis**
型 **number**

majorGrid

軸のグリッド線が表示されるかどうかを示す値を取得または設定します。

継承元 **Axis**
型 **boolean**

majorTickMarks

軸の目盛りマークの場所を取得または設定します。

継承元 **Axis**
型 **TickMark**

majorUnit

軸ラベル間の単位数を取得または設定します。

軸の値が日付の場合、単位は日数になります。

継承元 **Axis**
型 **number**

max

軸に表示される最大値を取得または設定します。

値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

The default value for this property is **null**, which causes the chart to calculate the maximum value based on the data.

継承元 **Axis**
型 **any**

min

軸に表示される最小値を取得または設定します。

値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

The default value for this property is **null**, which causes the chart to calculate the minimum value based on the data.

継承元 **Axis**
型 **any**

minorGrid

軸の副グリッド線が表示されるかどうかを示す値を取得または設定します。

継承元 **Axis**
型 **boolean**

minorTickMarks

小軸目盛りマークの場所を取得または設定します。

継承元 **Axis**
型 **TickMark**

minorUnit

軸の補助目盛間の単位数を取得または設定します。

軸の値が日付の場合、単位は日数になります。

継承元 **Axis**
型 **number**

name

軸の名前を取得または設定します。

継承元 **Axis**
型 **string**

origin

軸が直交軸と交差する位置の値を取得または設定します。

継承元 **Axis**
型 **number**

overlappingLabels

重なった軸ラベルの処理方法を示す値を取得または設定します。 The default value for this property is **OverlappingLabels.Auto**.

継承元 **Axis**
型 **OverlappingLabels**

plotArea

軸のプロットエリアを取得または設定します。

継承元 **Axis**
型 **PlotArea**

position

プロットエリアに対する軸の位置を取得または設定します。

継承元 **Axis**
型 **Position**

reversed

軸を反転させる（上から下方向、または右から左方向にする）かどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 **Axis**
型 **boolean**

title

軸の横に表示されるタイトルのテキストを取得または設定します。

継承元 **Axis**
型 **string**

メソッド

convert

```
convert(val: number, maxValue?: number, minValue?: number): number
```

指定した値をデータ座標からピクセル座標に変換します。

パラメーター

- **val**: **number**
  変換するデータ値。
- **maxValue**: **number** OPTIONAL
  データの最大値（オプション）。
- **minValue**: **number** OPTIONAL
  データの最小値（オプション）。

継承元 **Axis**
戻り値 **number**

convertBack

```
convertBack(val: number): number
```

指定した値をピクセル座標からデータ座標に変換します。

パラメーター

- **val**: **number**
  変換し戻すピクセル座標。

継承元 **Axis**
戻り値 **number**

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

rangeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Axis**
戻り値 **void**

イベント

rangeChanged

軸範囲が変更されたときに発生します。

継承元 **Axis**
引数 **EventArgs**

FlexRadarSeries クラス

ファイル	`wijmo.chart.radar.js`
モジュール	**wijmo.chart.radar**
基本クラス	**SeriesBase**
派生クラス	**WjFlexRadarSeries**
表示	☐継承されたメンバー ☐イベント発生元

チャートに表示するデータポイントの系列を表します。

FlexRadarSeries クラスは、基本的なチャートタイプのすべてをサポートします。**FlexRadar** 系列コレクションに追加する**FlexRadarSeries** オブジェクトごとに異なるチャートタイプを定義できます。これは、そのコレクション内のすべての**FlexRadarSeries** オブジェクトのデフォルトとしてチャートに設定された**chartType** プロパティをオーバーライドします。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): SeriesBase
```

SeriesBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **SeriesBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

chartType

特定の系列のチャートタイプを取得または設定します。これはチャート全体に設定されたチャートタイプをオーバーライドします。ColumnVolume、EquiVolume、CandleVolume、ArmsCandleVolumeの各チャートタイプはサポートされていない点に注意してください。これらのチャートタイプは**FinancialChart** に対して設定する必要があります

型 **RadarChartType**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

RadarChartType 列挙体

ファイル `wijmo.chart.radar.js`
モジュール **wijmo.chart.radar**

レーダーチャートのタイプを指定します。

メンバー

名前	値	説明
Column	0	縦棒を表示して、カテゴリ間で項目の値を比較できるようにします。
Scatter	1	X座標とY座標を使用して、データに含まれるパターンを表示します。
Line	2	一定期間のトレンドまたはカテゴリ間のトレンドを表示します。
LineSymbols	3	各データポイントにシンボルを使用する折れ線グラフを表示します。
Area	4	線の下の領域が色で塗りつぶされた折れ線グラフを表示します。

wijmo.gauge モジュール

ファイル `wijmo.gauge.js`
モジュール **wijmo.gauge**

RadialGauge 、**LinearGauge** 、**BulletGraph** の各コントロールを定義します。

多くのゲージコントロールとは異なり、Wijmoゲージは表示されるデータに焦点を合わせており、本質的に無関係な色要素やマークアップ要素はほとんどありません。これらのゲージは、特に画面の小さいデバイスでの使いやすさと読みやすさを重視して設計されています。

Wijmoゲージは**Range** オブジェクトで構成されています。すべてのWijmoゲージには、少なくとも"face"と"pointer"という2つの範囲があります。

- "face"は、ゲージの背景を表します。フェース範囲のminプロパティとmaxプロパティはゲージコントロールのminプロパティとmaxプロパティに対応し、ゲージに表示できる値を制限します。
- "pointer"は、ゲージの現在の値を示す範囲です。ポインタ範囲のmaxプロパティは、ゲージのvalueプロパティに対応します。

これら2つの特殊範囲に加えて、任意の数の追加範囲をゲージのrangesコレクションに追加できます。これらの追加範囲は以下の2つのことに使用できます。

- デフォルトでは、追加範囲はゲージの背景の一部として表示されます。 これにより、ゲージ内のゾーン（たとえば、良い、平均、悪い、など）を示すことができます。
- ゲージのshowRangesプロパティをfalseに設定した場合、追加範囲は表示されません。その代わりに、現在の値に基づいてポインタ範囲の色を自動的に設定するために使用されます。

クラス

- BulletGraph
- Gauge
- LinearGauge
- RadialGauge
- Range

列挙体

- GaugeDirection
- ShowText

BulletGraph クラス

ファイル wijmo.gauge.js
モジュール **wijmo.gauge**
基本クラス **LinearGauge**
派生クラス **WjBulletGraph**
表示 ☐継承されたメンバー ☐イベント発生元

BulletGraph は、ダッシュボード専用に設計された線形ゲージの一種です。単一の主要指標を比較指標や定性的範囲とともに表示するため、指標が良好、不良、またはその他の状態のいずれにあるかがすぐにわかります。

ブレットグラフは、ダッシュボード設計の専門家であるStephen Few氏が開発し、普及させました。詳細と具体例については、**Wikipedia** を参照してください。

サンプル

Show me (https://jsfiddle.net/Wijmo5/vqrwdvgq)

コンストラクタ

- constructor

プロパティ

- bad
- controlTemplate
- direction
- face
- format
- getText
- good
- hasShadow
- hostElement
- isAnimated
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- stackRanges
- step
- target
- thickness
- thumbSize
- tickSpacing
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): BulletGraph
```

BulletGraph クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **BulletGraph**

プロパティ

bad

指標が不良と見なされる基準値を取得または設定します。

型 **number**

STATIC controlTemplate

Gauge コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **Gauge**
型 **any**

direction

ゲージの値が増加する方向を取得または設定します。 The default value for this property is **GaugeDirection.Right**.

継承元 **LinearGauge**
型 **GaugeDirection**

face

ゲージの全体的な形状と外観を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

format

ゲージ値をテキストとして表示するために使用される書式文字列を取得または設定します。

継承元 **Gauge**
型 **string**

getText

ゲージ値の表示に使用されるカスタマイズされた文字列を返す コールバックを取得または設定します。

このプロパティは、ゲージに表示される文字列をカスタマイズしたいが、 **`format`** プロパティでは十分でない場合に使用してください。

指定する場合、コールバック関数は、ゲージ、部分名、値、および 書式設定された値をパラメータとして取る必要があります。また、ゲージに表示される文字列を返す必要があります。

次に例を示します。

```
// 値を文字列に変換するコールバック
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return '空！';
      if (value <= 25) return '低量...';
      if (value <= 95) return '適量';
      return '満タン';
    case 'min':
      return '空';
    case 'max':
      return '満タン';
  }
  return text;
}
```

継承元 **Gauge**
型 **Function**

good

指標が良好と見なされる基準値を取得または設定します。

型 **number**

hasShadow

ゲージに影の効果を表示するかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **`Control`**
型 **HTMLElement**

isAnimated

Gauge が値の変更を示すためにアニメーションを使用するかどうかを決定する値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用して値を編集できるかどうかを示す値を 取得または設定します。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- max

ゲージに表示できる最大値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- min

ゲージに表示できる最小値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- origin

範囲を描画するための始点を取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、値の範囲はゲージの最小値から始まります。最小値がゼロ未満の場合はゼロから始まります。

継承元 **Gauge**
型 **number**

- pointer

ゲージの現在の値を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

- ranges

このゲージの範囲のコレクションを取得します。

継承元 **Gauge**
型 **ObservableArray**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- showRanges

ranges プロパティに含まれる範囲をゲージに表示するかどうかを示す値を取得または設定します。

このプロパティがfalseに設定されている場合、**ranges** プロパティに含まれる範囲はゲージに表示されません。 代わりに、これらは、値の変化のアニメーション中に、**pointer** 範囲の色を補間するために使用されます。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

- showText

ゲージにテキストとして表示する**ShowText** の値を取得または設定します。 The default value for this property is **ShowText.All** for **RadialGauge** controls, and to **ShowText.None** for **LinearGauge** controls.

継承元 **Gauge**
型 **ShowText**

showTicks

ゲージで、各**step** 値に目盛りマークを表示するかどうかを決定するプロパティを 取得または設定します。

目盛りマークは、CSSで**wj-gauge**クラス名と**wj-ticks**クラス名を 使用して書式設定できます。次に例を示します。

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

The default value for this property is **false**.

継承元 **Gauge**
型 **boolean**

stackRanges

ranges コレクションに含まれる範囲をゲージ内に積み重ねるかどうかを決定する値を取得または設定します。

stackRanges のデフォルト値はfalseであり、**ranges** コレクションの範囲はゲージのフェイスと同じ太さで表示されます。**stackRanges** をtrueに設定すると、範囲が狭くなり、並んで表示されます。

継承元 **Gauge**
型 **boolean**

step

ユーザーが方向キーを押すかマウスホイールを回したときに**value** プロパティを増減する量を取得または設定します。

継承元 **Gauge**
型 **number**

target

指標の目標値を取得または設定します。

型 **number**

thickness

ゲージの太さを0~1のスケールで取得または設定します。

thicknessを1に設定すると、ゲージは最大限に太くなります。値が小さいほどゲージは細くなります。

継承元 **Gauge**
型 **number**

thumbSize

ゲージの現在の値を示す要素のサイズ（ピクセル単位）を取得または設定します。

継承元 **Gauge**
型 **number**

tickSpacing

目盛りマークの間隔を取得または設定します。

ゲージ上に目盛りを表示するようにするには、**showTicks** プロパティをtrueに設定します。デフォルトでは、目盛りマークの間隔は**step** プロパティによって定義されます。

tickSpacing プロパティを使用してデフォルトをオーバーライドし、 **step** プロパティの値と異なるスペーシングを設定できます。デフォルトの動作に戻すには、**tickSpacing** プロパティをnullに設定します。

継承元 **Gauge**
型 **number**

value

ゲージに表示される値を取得または設定します。

継承元 **Gauge**
型 **number**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): number
```

指定したポイントにあるゲージの値に対応する数値を取得します。

例:

```
// ユーザーがゲージをクリックしたときにそのポイントのヒットテストを行います。
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）、MouseEvent オブジェクト、またはポイントのX座標。
- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **Gauge**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Gauge**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コン トロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示します。

継承元 **Gauge**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

value プロパティの値が変更されたときに発生します。

継承元 **Gauge**
引数 **EventArgs**

Gauge クラス

ファイル wijmo.gauge.js
モジュール **wijmo.gauge**
基本クラス **Control**
派生クラス **LinearGauge**, **RadialGauge**
表示 ☐継承されたメンバー ☐イベント発生元

Wijmo Gaugeコントロールの基本クラス（抽象）。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- face
- format
- getText
- hasShadow
- hostElement
- isAnimated
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- stackRanges
- step
- thickness
- thumbSize
- tickSpacing
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus

- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): Gauge
```

Gauge クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **Gauge**

プロパティ

- STATIC controlTemplate

Gauge コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

- face

ゲージの全体的な形状と外観を表すために使用される**Range** を取得または設定します。

型 **Range**

format

ゲージ値をテキストとして表示するために使用される書式文字列を取得または設定します。

型 **string**

getText

ゲージ値の表示に使用されるカスタマイズされた文字列を返す コールバックを取得または設定します。

このプロパティは、ゲージに表示される文字列をカスタマイズしたいが、 **`format`** プロパティでは十分でない場合に使用してください。

指定する場合、コールバック関数は、ゲージ、部分名、値、および 書式設定された値をパラメータとして取る必要があります。また、ゲージに表示される文字列を返す必要があります。

次に例を示します。

```
// 値を文字列に変換するコールバック
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return '空！';
      if (value <= 25) return '低量...';
      if (value <= 95) return '適量';
      return '満タン';
    case 'min':
      return '空';
    case 'max':
      return '満タン';
  }
  return text;
}
```

型 **Function**

hasShadow

ゲージに影の効果を表示するかどうかを示す値を取得または設定します。 The default value for this property is **true**.

型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **`Control`**
型 **HTMLElement**

isAnimated

Gauge が値の変更を示すためにアニメーションを使用するかどうかを決定する値を取得または設定します。 The default value for this property is **true**.

型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用して値を編集できるかどうかを示す値を 取得または設定します。

The default value for this property is **true**.

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- max

ゲージに表示できる最大値を取得または設定します。

`min` および**`max`** プロパティの使用方法については、 「**`min`および`max`プロパティの使用**」トピックを参照してください。

型 **number**

- min

ゲージに表示できる最小値を取得または設定します。

`min` および**`max`** プロパティの使用方法については、 「**`min`および`max`プロパティの使用**」トピックを参照してください。

型 **number**

- origin

範囲を描画するための始点を取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、値の範囲はゲージの最小値から始まります。最小値がゼロ未満の場合はゼロから始まります。

型 **number**

- pointer

ゲージの現在の値を表すために使用される**Range** を取得または設定します。

型 **Range**

- ranges

このゲージの範囲のコレクションを取得します。

型 **ObservableArray**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- showRanges

ranges プロパティに含まれる範囲をゲージに表示するかどうかを示す値を取得または設定します。

このプロパティがfalseに設定されている場合、**ranges** プロパティに含まれる範囲はゲージに表示されません。 代わりに、これらは、値の変化のアニメーション中に、**pointer** 範囲の色を補間するために使用されます。

The default value for this property is **true**.

型 **boolean**

- showText

ゲージにテキストとして表示する**ShowText** の値を取得または設定します。 The default value for this property is **ShowText.All** for **RadialGauge** controls, and to **ShowText.None** for **LinearGauge** controls.

型 **ShowText**

- showTicks

ゲージで、各**step** 値に目盛りマークを表示するかどうかを決定するプロパティを 取得または設定します。

目盛りマークは、CSSで**wj-gauge**クラス名と**wj-ticks**クラス名を 使用して書式設定できます。次に例を示します。

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

The default value for this property is **false**.

型 **boolean**

- stackRanges

ranges コレクションに含まれる範囲をゲージ内に積み重ねるかどうかを決定する値を取得または設定します。

stackRanges のデフォルト値はfalseであり、**ranges** コレクションの範囲はゲージのフェイスと同じ太さで表示されます。**stackRanges** をtrueに設定すると、範囲が狭くなり、並んで表示されます。

型 **boolean**

- step

ユーザーが方向キーを押すかマウスホイールを回したときに**value** プロパティを増減する量を取得または設定します。

型 **number**

- thickness

ゲージの太さを0~1のスケールで取得または設定します。

thicknessを1に設定すると、ゲージは最大限に太くなります。値が小さいほどゲージは細くなります。

型 **number**

- thumbSize

ゲージの現在の値を示す要素のサイズ（ピクセル単位）を取得または設定します。

型 **number**

- tickSpacing

目盛りマークの間隔を取得または設定します。

ゲージ上に目盛りを表示するようにするには、**showTicks** プロパティをtrueに設定します。デフォルトでは、目盛りマークの間隔は**step** プロパティによって定義されます。

tickSpacing プロパティを使用してデフォルトをオーバーライドし、**step** プロパティの値と異なるスペーシングを設定できます。デフォルトの動作に戻すには、**tickSpacing** プロパティをnullに設定します。

型 **number**

- value

ゲージに表示される値を取得または設定します。

型 **number**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): number
```

指定したポイントにあるゲージの値に対応する数値を取得します。

例:

```
// ユーザーがゲージをクリックしたときにそのポイントのヒットテストを行います。
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）、MouseEvent オブジェクト、またはポイントのX座標。
- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コン トロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示します。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

value プロパティの値が変更されたときに発生します。

引数 **EventArgs**

LinearGauge クラス

ファイル wijmo.gauge.js
モジュール **wijmo.gauge**
基本クラス **Gauge**
派生クラス **BulletGraph**, **WjLinearGauge**
表示 ☐継承されたメンバー ☐イベント発生元

LinearGauge は、1つの値を表すインジケータと、参照値を表す範囲 （オプション）を使用して、線形目盛りを表示します。

ゲージの**isReadOnly** プロパティを falseに設定すると、ユーザーがゲージをクリックして値を編集できるようになります。

サンプル

Show me (https://jsfiddle.net/Wijmo5/wkcehhvu)

コンストラクタ

- constructor

プロパティ

- controlTemplate
- direction
- face
- format
- getText
- hasShadow
- hostElement
- isAnimated
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- stackRanges
- step
- thickness
- thumbSize
- tickSpacing
- value

メソッド

- addEventListener
- applyTemplate

- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): LinearGauge
```

LinearGauge クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **LinearGauge**

プロパティ

STATIC controlTemplate

Gauge コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **Gauge**
型 **any**

direction

ゲージの値が増加する方向を取得または設定します。 The default value for this property is **GaugeDirection.Right**.

型 **GaugeDirection**

face

ゲージの全体的な形状と外観を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

format

ゲージ値をテキストとして表示するために使用される書式文字列を取得または設定します。

継承元 **Gauge**
型 **string**

getText

ゲージ値の表示に使用されるカスタマイズされた文字列を返す コールバックを取得または設定します。

このプロパティは、ゲージに表示される文字列をカスタマイズしたいが、 **format** プロパティでは十分でない場合に使用してください。

指定する場合、コールバック関数は、ゲージ、部分名、値、および 書式設定された値をパラメータとして取る必要があります。また、ゲージに表示される文字列を返す必要があります。

次に例を示します。

```
// 値を文字列に変換するコールバック
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return '空！';
      if (value <= 25) return '低量...';
      if (value <= 95) return '適量';
      return '満タン';
    case 'min':
      return '空';
    case 'max':
      return '満タン';
  }
  return text;
}
```

継承元 **Gauge**
型 **Function**

hasShadow

ゲージに影の効果を表示するかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isAnimated

Gauge が値の変更を示すためにアニメーションを使用するかどうかを決定する値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用して値を編集できるかどうかを示す値を 取得または設定します。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- max

ゲージに表示できる最大値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- min

ゲージに表示できる最小値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- origin

範囲を描画するための始点を取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、値の範囲はゲージの最小値から始まります。最小値がゼロ未満の場合はゼロから始まります。

継承元 **Gauge**
型 **number**

- pointer

ゲージの現在の値を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

- ranges

このゲージの範囲のコレクションを取得します。

継承元 **Gauge**
型 **ObservableArray**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

showRanges

ranges プロパティに含まれる範囲をゲージに表示するかどうかを示す値を取得または設定します。

このプロパティがfalseに設定されている場合、**ranges** プロパティに含まれる範囲はゲージに表示されません。 代わりに、これらは、値の変化のアニメーション中に、**pointer** 範囲の色を補間するために使用されます。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

showText

ゲージにテキストとして表示する**ShowText** の値を取得または設定します。 The default value for this property is **ShowText.All** for **RadialGauge** controls, and to **ShowText.None** for **LinearGauge** controls.

継承元 **Gauge**
型 **ShowText**

showTicks

ゲージで、各**step** 値に目盛りマークを表示するかどうかを決定するプロパティを 取得または設定します。

目盛りマークは、CSSで**wj-gauge**クラス名と**wj-ticks**クラス名を 使用して書式設定できます。次に例を示します。

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

The default value for this property is **false**.

継承元 **Gauge**
型 **boolean**

stackRanges

ranges コレクションに含まれる範囲をゲージ内に積み重ねるかどうかを決定する値を取得または設定します。

stackRanges のデフォルト値はfalseであり、**ranges** コレクションの範囲はゲージのフェイスと同じ太さで表示されます。**stackRanges** をtrueに設定すると、範囲が狭くなり、並んで表示されます。

継承元 **Gauge**
型 **boolean**

step

ユーザーが方向キーを押すかマウスホイールを回したときに**value** プロパティを増減する量を取得または設定します。

継承元 **Gauge**
型 **number**

thickness

ゲージの太さを0~1のスケールで取得または設定します。

thicknessを1に設定すると、ゲージは最大限に太くなります。値が小さいほどゲージは細くなります。

継承元 **Gauge**
型 **number**

thumbSize

ゲージの現在の値を示す要素のサイズ（ピクセル単位）を取得または設定します。

継承元 **Gauge**
型 **number**

tickSpacing

目盛りマークの間隔を取得または設定します。

ゲージ上に目盛りを表示するようにするには、**showTicks** プロパティをtrueに設定します。デフォルトでは、目盛りマークの間隔は**step** プロパティによって定義されます。

tickSpacing プロパティを使用してデフォルトをオーバーライドし、**step** プロパティの値と異なるスペーシングを設定できます。デフォルトの動作に戻すには、**tickSpacing** プロパティをnullに設定します。

継承元 **Gauge**
型 **number**

value

ゲージに表示される値を取得または設定します。

継承元 **Gauge**
型 **number**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): number
```

指定したポイントにあるゲージの値に対応する数値を取得します。

例:

```
// ユーザーがゲージをクリックしたときにそのポイントのヒットテストを行います。
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）、MouseEvent オブジェクト、またはポイントのX座標。
- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **Gauge**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Gauge**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コン トロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示します。

継承元 **Gauge**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

value プロパティの値が変更されたときに発生します。

継承元 **Gauge**
引数 **EventArgs**

RadialGauge クラス

ファイル wijmo.gauge.js
モジュール **wijmo.gauge**
基本クラス **Gauge**
派生クラス **WjRadialGauge**
表示 ☐継承されたメンバー ☐イベント発生元

RadialGauge は、1つの値を表すインジケータと、参照値を表す範囲 （オプション）を使用して、円形目盛りを表示します。

ゲージの**isReadOnly** プロパティを falseに設定すると、ユーザーがゲージをクリックして値を編集できるようになります。

サンプル

Show me (https://jsfiddle.net/Wijmo5/kqkm8zt0)

コンストラクタ

- constructor

プロパティ

- autoScale
- clientSize
- controlTemplate
- face
- format
- getText
- hasShadow
- hostElement
- isAnimated
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- stackRanges
- startAngle
- step
- sweepAngle
- thickness
- thumbSize
- tickSpacing
- value

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): RadialGauge
```

RadialGauge クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **RadialGauge**

プロパティ

autoScale

ゲージがホスト要素に収まるように自動的に拡大縮小されるかどうかを示す値を取得または設定します The default value for this property is **true**.

型 **boolean**

clientSize

autoScale、パディング、枠線、および余白を考慮して、ゲージのクライアント領域のサイズを取得します。

型 **Size**

STATIC controlTemplate

Gauge コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **Gauge**
型 **any**

face

ゲージの全体的な形状と外観を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

format

ゲージ値をテキストとして表示するために使用される書式文字列を取得または設定します。

継承元 **Gauge**
型 **string**

getText

ゲージ値の表示に使用されるカスタマイズされた文字列を返す コールバックを取得または設定します。

このプロパティは、ゲージに表示される文字列をカスタマイズしたいが、 **format** プロパティでは十分でない場合に使用してください。

指定する場合、コールバック関数は、ゲージ、部分名、値、および 書式設定された値をパラメータとして取る必要があります。また、ゲージに表示される文字列を返す必要があります。

次に例を示します。

```
// 値を文字列に変換するコールバック
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return '空！';
      if (value <= 25) return '低量...';
      if (value <= 95) return '適量';
      return '満タン';
    case 'min':
      return '空';
    case 'max':
      return '満タン';
  }
  return text;
}
```

継承元 **Gauge**
型 **Function**

hasShadow

ゲージに影の効果を表示するかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isAnimated

Gauge が値の変更を示すためにアニメーションを使用するかどうかを決定する値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用して値を編集できるかどうかを示す値を 取得または設定します。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- max

ゲージに表示できる最大値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- min

ゲージに表示できる最小値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- origin

範囲を描画するための始点を取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、値の範囲はゲージの最小値から始まります。最小値がゼロ未満の場合はゼロから始まります。

継承元 **Gauge**
型 **number**

- pointer

ゲージの現在の値を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

ranges

このゲージの範囲のコレクションを取得します。

継承元 **Gauge**
型 **ObservableArray**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

showRanges

ranges プロパティに含まれる範囲をゲージに表示するかどうかを示す値を取得または設定します。

このプロパティがfalseに設定されている場合、**ranges** プロパティに含まれる範囲はゲージに表示されません。 代わりに、これらは、値の変化のアニメーション中に、**pointer** 範囲の色を補間するために使用されます。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

showText

ゲージにテキストとして表示する**ShowText** の値を取得または設定します。 The default value for this property is **ShowText.All** for **RadialGauge** controls, and to **ShowText.None** for **LinearGauge** controls.

継承元 **Gauge**
型 **ShowText**

showTicks

ゲージで、各**step** 値に目盛りマークを表示するかどうかを決定するプロパティを 取得または設定します。

目盛りマークは、CSSで**wj-gauge**クラス名と**wj-ticks**クラス名を 使用して書式設定できます。次に例を示します。

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

The default value for this property is **false**.

継承元 **Gauge**
型 **boolean**

stackRanges

ranges コレクションに含まれる範囲をゲージ内に積み重ねるかどうかを決定する値を取得または設定します。

stackRanges のデフォルト値はfalseであり、**ranges** コレクションの範囲はゲージのフェイスと同じ太さで表示されます。**stackRanges** をtrueに設定すると、範囲が狭くなり、並んで表示されます。

継承元 **Gauge**
型 **boolean**

startAngle

ゲージの開始角度（度単位）を取得または設定します。

角度は9時の位置から時計回りに度単位で測定されます。

The default value for this property is **0**.

型 **number**

step

ユーザーが方向キーを押すかマウスホイールを回したときに**value** プロパティを増減する量を取得または設定します。

継承元 **Gauge**
型 **number**

sweepAngle

ゲージの掃引角度（度単位）を取得または設定します。

角度は9時の位置から時計回りに度単位で測定されます。

The default value for this property is **180**.

型 **number**

thickness

ゲージの太さを0~1のスケールで取得または設定します。

thicknessを1に設定すると、ゲージは最大限に太くなります。値が小さいほどゲージは細くなります。

継承元 **Gauge**
型 **number**

thumbSize

ゲージの現在の値を示す要素のサイズ（ピクセル単位）を取得または設定します。

継承元 **Gauge**
型 **number**

tickSpacing

目盛りマークの間隔を取得または設定します。

ゲージ上に目盛りを表示するようにするには、**showTicks** プロパティをtrueに設定します。デフォルトでは、目盛りマークの間隔は**step** プロパティによって定義されます。

tickSpacing プロパティを使用してデフォルトをオーバーライドし、 **step** プロパティの値と異なるスペーシングを設定できます。デフォルトの動作に戻すには、**tickSpacing** プロパティをnullに設定します。

継承元 **Gauge**
型 **number**

value

ゲージに表示される値を取得または設定します。

継承元 **Gauge**
型 **number**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): number
```

指定したポイントにあるゲージの値に対応する数値を取得します。

例:

```
// ユーザーがゲージをクリックしたときにそのポイントのヒットテストを行います。
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）、MouseEvent オブジェクト、またはポイントのX座標。
- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **Gauge**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Gauge**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示します。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

value プロパティの値が変更されたときに発生します。

継承元 **Gauge**
引数 **EventArgs**

Range クラス

ファイル wijmo.gauge.js
モジュール **wijmo.gauge**
派生クラス **WjRange**

Gauge コントロールで使用される範囲を定義します。

Range オブジェクトには、範囲の境界を定義する**min** および**max** プロパティと、範囲の外観を定義する**color** および**thickness** プロパティがあります。

どの**Gauge** コントロールにも、少なくとも次の2つの範囲があります。 'face'範囲はゲージの最小値と最大値を定義し、 'pointer'範囲はゲージの現在の値を表示します。

これらの組み込み範囲に加えて、重要な領域（たとえば、低い値、中間の値、高い値など）の表示に使用される追加の範囲をゲージに設定できます。

コンストラクタ

- constructor

プロパティ

- color
- max
- min
- name
- thickness

メソッド

- onPropertyChanged

イベント

- propertyChanged

コンストラクタ

constructor

```
constructor(options?: string): Range
```

Range クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **string** OPTIONAL
  Range 、または**Range** の名前を含む文字列の初期化オプション。

戻り値 **Range**

プロパティ

- color

この範囲の表示に使用される色を取得または設定します。

型 **string**

max

この範囲の最大値を取得または設定します。

型 **number**

min

この範囲の最小値を取得または設定します。

型 **number**

name

この**Range** の名前を取得または設定します。

型 **string**

thickness

この範囲の太さを親ゲージの太さの割合（%）として取得または設定します。

型 **number**

メソッド

onPropertyChanged

```
onPropertyChanged(e: PropertyChangedEventArgs): void
```

propertyChanged イベントを発生させます。

パラメーター

- **e**: **PropertyChangedEventArgs**
  プロパティ名および新旧の値を含む**PropertyChangedEventArgs** 。

戻り値 **void**

イベント

propertyChanged

この**Range** のプロパティ値が変更されると発生します。

引数 **PropertyChangedEventArgs**

GaugeDirection 列挙体

ファイル `wijmo.gauge.js`
モジュール **wijmo.gauge**

LinearGauge のポインタが増加する方向を表します。

メンバー

名前	値	説明
Right	0	ゲージの値は左から右へ増加します。
Left	1	ゲージの値は右から左へ増加します。
Up	2	ゲージの値は下から上へ増加します。
Down	3	ゲージの値は上から下へ増加します。

ShowText 列挙体

ファイル `wijmo.gauge.js`
モジュール **wijmo.gauge**

どの値をテキストとして表示するかを指定します。

メンバー

名前	値	説明
None	0	ゲージにテキストを表示しません。
Value	1	ゲージの**`value`** をテキストとして表示します。
MinMax	2	ゲージの**`min`** 値と**`max`** 値をテキストとして表示します。
All	3	ゲージの**`value`** 、**`min`** 、および**`max`** をテキストとして表示します。

wijmo.odata モジュール

ファイル `wijmo.odata.js`
モジュール **wijmo.odata**

ODataプロトコルをサポートするクラス（**ODataCollectionView** クラスなど）を提供します。

ODataは、データAPIを作成および使用するための標準プロトコルです。 ODataは、HTTPなどのコアプロトコルとRESTなどの一般に認められた方法論を基に構築されています。 その結果、完全な機能を備えたデータAPIが統一的に公開されています。(http://www.odata.org/) (http://www.odata.org/)

クラス

- ODataCollectionView
- ODataVirtualCollectionView

ODataCollectionView クラス

ファイル wijmo.odata.js
モジュール **wijmo.odata**
基本クラス **CollectionView**
派生クラス **ODataVirtualCollectionView**
表示 ☐継承されたメンバー ☐イベント発生元

CollectionView クラスを拡張して、ODataソースからのデータのロードおよびODataソースへの データの保存をサポートします。

ODataCollectionView クラスを使用して、ODataサービスからデータをロードし、 そのデータを任意のWijmoコントロールのデータソースとして使用できます。

完全なCRUDサポートに加えて、ソート、フィルタ処理、ページング、グループ化などの **CollectionView** 機能を使用できます。 ソート、フィルタ処理、および ページング機能は、サーバーでもクライアントでも実行できます。

以下のコードは、データソースからいくつかのフィールドを選択し、クライアントにソート機能を提供する **ODataCollectionView** を インスタンス化する方法を示します。 初期化データを渡すために'options'パラメータが使用されていますが、これはコントロールの初期化と同じ手法です。

```
var url = 'http://services.odata.org/Northwind/Northwind.svc';
var categories = new wijmo.odata.ODataCollectionView(url, 'Categories', {
  fields: ['CategoryID', 'CategoryName', 'Description'],
  sortOnServer: false
});
```

次の例では、**ODataCollectionView** を使用してNorthWind ODataプロバイダサービスからデータをロードし、その結果を **FlexGrid** コントロールに表示する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/r5a21ysm)

コンストラクタ

- constructor

プロパティ

- canAddNew
- canCancelEdit
- canChangePage
- canFilter
- canGroup
- canRemove
- canSort
- currentAddItem
- currentEditItem
- currentItem
- currentPosition
- dataTypes
- expand
- fields
- filter
- filterDefinition
- filterOnServer
- getError
- groupDescriptions

- groups
- inferDataTypes
- isAddingNew
- isEditingItem
- isEmpty
- isLoading
- isPageChanging
- isUpdating
- itemCount
- items
- itemsAdded
- itemsEdited
- itemsRemoved
- jsonReviver
- keys
- newItemCreator
- oDataVersion
- pageCount
- pageIndex
- pageOnServer
- pageSize
- requestHeaders
- showDatesAsGmt
- sortComparer
- sortConverter
- sortDescriptions
- sortNullsFirst
- sortOnServer
- sourceCollection
- tableName
- totalItemCount
- trackChanges
- useStableSort

メソッド

- addNew
- beginUpdate
- cancelEdit
- cancelNew
- clearChanges
- commitEdit
- commitNew
- contains
- deferUpdate
- editItem
- endUpdate
- getAggregate
- implementsInterface
- load

- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage
- onCollectionChanged
- onCurrentChanged
- onCurrentChanging
- onError
- onLoaded
- onLoading
- onPageChanged
- onPageChanging
- onSourceCollectionChanged
- onSourceCollectionChanging
- refresh
- remove
- removeAt
- updateFilterDefinition

イベント

- collectionChanged
- currentChanged
- currentChanging
- error
- loaded
- loading
- pageChanged
- pageChanging
- sourceCollectionChanged
- sourceCollectionChanging

コンストラクタ

constructor

```
constructor(url: string, tableName: string, options?: any): ODataCollectionView
```

ODataCollectionView クラスの新しいインスタンスを初期化します。

パラメーター

- **url**: **string**
  ODataサービスのURL（たとえば、'http://services.odata.org/Northwind/Northwind.svc'）。
- **tableName**: **string**
  サービスから取得するテーブル（エンティティ）の名前。指定されない場合は、有効なテーブル（エンティティ）のリストが取得されます。
- **options**: **any** OPTIONAL
  ODataCollectionView の初期化データ（プロパティ値とイベントハンドラ）を含むJavaScriptオブジェクト。

戻り値 **ODataCollectionView**

プロパティ

canAddNew

コレクションに新しい項目を追加できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canCancelEdit

適用前の変更を破棄して編集されたオブジェクトの元の値を復元できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canChangePage

pageIndex 値を変更できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canFilter

このビューが**filter** プロパティによってフィルタリングをサポートしているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canGroup

このビューが**groupDescriptions** プロパティによってグループ化をサポートしているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canRemove

コレクションから項目を削除できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canSort

このビューが**sortDescriptions** プロパティによってソートをサポートしているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

currentAddItem

現在の追加トランザクションの間に追加される項目を取得します。

継承元 **CollectionView**
型 **any**

currentEditItem

現在の編集トランザクションの間に編集される項目を取得します。

継承元 **CollectionView**
型 **any**

currentItem

ビューの現在の項目を取得します。

継承元 **CollectionView**
型 **any**

currentPosition

ビューの現在の項目の順序位置を取得します。

継承元 **CollectionView**
型 **number**

dataTypes

データのロード中にデータ型を変換するためのマップとして使用されるJavaScriptオブジェクトを取得または設定します。

オブジェクトのキーはフィールド名を表し、値はそのデータをどのように型変換するかを示す**DataType** 値を表します。

以下のサンプルコードは、**ODataCollectionView** を作成した後、（データベースに文字列として格納されている）'Freight'の 値を数値に変換し、さらに3つの日付フィールドを日付に変換します。

```
var orders = new wijmo.data.ODataCollectionView(url, 'Orders', {
  dataTypes: {
    Freight: wijmo.DataType.Number
    OrderDate: wijmo.DataType.Date,
    RequiredDate: wijmo.DataType.Date,
    ShippedDate: wijmo.DataType.Date,
  }
});
```

このプロパティは、データベースに格納されているデータの書式が一般的な用法に適合していない場合に役立ちます。

inferDataTypes プロパティによってDate値の変換が自動的に処理されるため、ほとんどの場合、データ型に関する情報を提供する必要はありません。

明示的に型情報を提供した場合、**inferDataTypes** プロパティは適用されません。そのため、データ型の情報を提供する場合は、Date型 のフィールドも含め、すべてのフィールドの型情報を提供する必要があります。

型　　**any**

expand

関連エンティティを返すデータに含めるかどうかを指定する文字列を取得または設定します。

このプロパティは直接ODataの$expandオプションにマップされます。

たとえば、次のコードは、すべての顧客とその注文をデータベースから取得します。各顧客エンティティには、注文オブジェクトの配列を含む[Orders]フィールドが含まれています。

```
var url = 'http://services.odata.org/Northwind/Northwind.svc';
var customersOrders = new wijmo.odata.ODataCollectionView(url, 'Customers', {
  expand: 'Orders'
});
```

型　　**string**

fields

データソースから取得するフィールドの名前を含む配列を取得または設定します。

このプロパティがnullまたは空の配列に設定されている場合は、すべてのフィールドが 取得されます。

たとえば、以下のコードは、データベースの'Categories'テーブルから3つの フィールドのみを取得する **ODataCollectionView** を作成します。

```
var categories = new wijmo.data.ODataCollectionView(url, 'Categories', {
  fields: ['CategoryID', 'CategoryName', 'Description']
});
```

型　　**string**[]

filter

項目がビューに含める対象として適しているかどうかを判断するために使用されるコールバックを取得または設定します。

このコールバック関数がtrueを返した場合、パラメーターとして渡された項目はビューに含まれます。

メモ: フィルタ関数でスコープ（すなわち、有効な'this'値）が必要な場合は、'this'オブジェクトを指定した'bind'関数を使用してフィルタを設定します。例:

```
collectionView.filter = this._filter.bind(this);
```

継承元 **CollectionView**
型 **IPredicate**

filterDefinition

サーバーでデータをフィルタ処理するために使用されるODataフィルタ仕様 を含む文字列を取得または設定します。

フィルタ定義構文については、 **ODataドキュメント**を参照してください。

たとえば、以下のコードでは、'CompanyName'フィールドが'A'で始まり'S'で終わる レコードがサーバーから返されます。

```
view.filterDefinition = "startswith(CompanyName, 'A') and endswith(CompanyName, 'B')";
```

フィルタ定義は自動生成することができます。たとえば、 **FlexGridFilter** コンポーネントは、データソースが **ODataCollectionView** かどうかを検出し、**filter** プロパティと **filterDefinition** プロパティの両方を自動的に更新します。

filterDefinition プロパティは、**filterOnServer** プロパティが falseに設定されている場合でも適用されることに注意してください。 これにより、同じコレクションに対してサーバーフィルタとクライアントフィルタの両方を適用でき、さまざまなシナリオで役立ちます。

たとえば、次のコードは、**filterDefinition** プロパティを使用して サーバーでフィルタ処理を実行し、 **filter** プロパティを使用して クライアントでさらにフィルタ処理を実行します。結果のコレクションには、名前が'C'で始まり、単価が20より 大きい項目が表示されます。

```
var url = 'http://services.odata.org/V4/Northwind/Northwind.svc/';
var data = new wijmo.odata.ODataCollectionView(url, 'Products', {
  oDataVersion: 4,
  filterDefinition: 'startswith(ProductName, \'C\')', // サーバー側のフィルタ
  filterOnServer: false, // クライアント側のフィルタ
  filter: function(product) {
    return product.UnitPrice > 20;
  },
});
```

型 **string**

filterOnServer

フィルタリングがサーバー側とクライアント側のどちらで実行されるかを決定する値を取得または設定します。

フィルタリングをクライアント側で実行する場合は**filter** プロパティを使用し、サーバー側で実行する場合は**filterDefinition** プロパティを使用します。

場合によっては、クライアント側とサーバー側の**両方**で個別にフィルタを適用したいことがあります。

そのためには、（1）filterOnServerプロパティをfalseに設定して**filter** プロパティにフィルタ関数を設定し （これでクライアント側フィルタリングが有効になります）、（2）**filterDefinition** プロパティにフィルタ 文字列を設定します（これでサーバー側フィルタリングが有効になります）。

The default value for this property is **true**.

型 **boolean**

getError

項目の特定のプロパティに検証エラーが含まれているかどうかを判定するコールバックを取得または設定します。

指定すると、コールバックは、検証する項目とプロパティを含む2つの パラメータを受け取り、エラーを説明する文字列を返します （エラーがない場合はnull）。

次に例を示します。

```
var view = new wijmo.collections.CollectionView(data, {
    getError: function (item, property) {
        switch (property) {
            case 'country':
                return countries.indexOf(item.country) < 0
                    ? 'Invalid Country'
                    : null;
            case 'downloads':
            case 'sales':
            case 'expenses':
                return item[property] < 0
                    ? '負にはできません。'
                    : null;
            case 'active':
                return item.active && item.country.match(/US|UK/)
                    ? 'USまたはUKではアクティブな項目が許可されません。'
                    : null;
        }
        return null;
    }
});
```

継承元 **CollectionView**
型 **Function**

groupDescriptions

コレクションの項目をビューでどのようにグループ化するかを記述する **GroupDescription** オブジェクトのコレクションを取得します。

継承元 **CollectionView**
型 **ObservableArray**

groups

最上位レベルのグループを表す **CollectionViewGroup** オブジェクトの配列を取得します。

継承元 **CollectionView**
型 **CollectionViewGroup**[]

inferDataTypes

標準の日付表現のような文字列を含むフィールドを 自動的に日付に変換するかどうかを決定する 値を取得または設定します。

ODataCollectionView クラスは、DateオブジェクトをサポートしていないJSON形式を 使用するため、このプロパティはデフォルトでtrueに設定されます。

dataTypes プロパティを使用して特定のタイプ情報を指定する場合、 このプロパティは無効になります。

型 **boolean**

- isAddingNew

追加トランザクションが進行中であるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isEditingItem

編集トランザクションが進行中であるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isEmpty

このビューに項目が1つも含まれていないかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isLoading

ODataCollectionView が現在データをロードしていることを示す 値を取得します。

このプロパティを使用して、進捗状況インジケータを提供できます。

型 **boolean**

- isPageChanging

ページインデックスが変更されているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

継承元 **CollectionView**
型

- itemCount

ページングを適用する前のビューの既知の項目の数を取得します。

継承元 **CollectionView**
型 **number**

- items

ビューの項目を取得します。

継承元 **CollectionView**
型 **any**[]

- itemsAdded

trackChanges が有効化されてから、コレクションに追加されたレコードを含む**ObservableArray** を取得します。

継承元 **CollectionView**
型 **ObservableArray**

- itemsEdited

trackChanges が有効化されてから、コレクションで編集されたレコードを含む**ObservableArray** を取得します。

継承元 **CollectionView**
型 **ObservableArray**

- itemsRemoved

trackChanges が有効化されてから、コレクションから削除されたレコードを含む**ObservableArray** を取得します。

継承元 **CollectionView**
型 **ObservableArray**

- jsonReviver

サーバーから返されるJSON値を解析するときに使用するカスタムのreviver関数を取得または設定します。

提供されている場合、関数は2つのパラメータ（キーおよび値）があり、解析された値（元の値と同じ値の場合があります）を返さなければなりません。

reviver関数の詳細については、 **JSON.parse メソッド**を参照してください。。

型 **Function**

- keys

キーフィールドの名前を含む配列を取得または設定します。

更新操作（追加/削除/完全削除）にはキーフィールドが必要です。

型 **string**[]

- newItemCreator

コレクションの新しい項目を作成する関数を取得または設定します。

作成関数が提供されない場合、**CollectionView** は、 適切な型の項目を初期化せずに作成しようとします。

作成関数が提供される場合、その関数は、パラメータを受け取らず、 コレクションに対して適切な型のオブジェクトを初期化して返す 必要があります。

継承元 **CollectionView**
型 **Function**

oDataVersion

サーバーによって使用されるODataのバージョンを取得または設定します。

現在、ODataサービスには1.0〜4.0の4つのバージョンがあります。 最新サービスではバージョン4.0が使用されていますが、レガシーサービスも依然として数多く運用されています。

利用するサービスが実装しているODataのバージョンがわかっている場合は、**ODataCollectionView** を作成するときに、 **oDataVersion** プロパティを適切な値（1〜4）に設定します（以下の例を参照）。

```
var url = 'http://services.odata.org/Northwind/Northwind.svc';
var categories = new wijmo.odata.ODataCollectionView(url, 'Categories', {
  oDataVersion: 1.0, // legacy OData source
  fields: ['CategoryID', 'CategoryName', 'Description'],
  sortOnServer: false
});;
```

利用するサービスが実装しているODataのバージョンがわからない場合（おそらく、 ODataエクスプローラアプリケーションを記述している場合）には、 バージョンは指定しないでください。その場合、**ODataCollectionView** この情報をサーバーから取得します。この操作には追加のリクエストが必要ですが、サービスURLあたり1回だけなので、オーバーヘッドは大きくありません。

型 **number**

pageCount

総ページ数を取得します。

型 **number**

pageIndex

現在のページの0から始まるインデックスを取得します。

継承元 **CollectionView**
型 **number**

pageOnServer

ページングをサーバーで実行するか、クライアントで実行するかを決定する 値を取得または設定します。

ページングを有効にするには、**pageSize** プロパティを使用します。

The default value for this property is **true**.

型 **boolean**

pageSize

1ページに表示する項目数を取得または設定します。

型 **number**

requestHeaders

データを送信または要求するときに使用する要求ヘッダーを含むオブジェクトを取得または設定します。

このプロパティは、認証が必要なシナリオでよく使用されます。次に例を示します。

```
var categories = new wijmo.odata.ODataCollectionView(serviceUrl, 'Categories', {
  fields: ['Category_ID', 'Category_Name'],
  requestHeaders: { Authorization: db.token }
});
```

型 **any**

showDatesAsGmt

日付を現地日ではなくGMTのように調整するかどうかを決定する値を取得または設定します。

型

sortComparer

ソート時に値の比較に使用する関数を取得または設定します。

指定された場合、ソート比較関数は、パラメータとして任意の型の値を 2つ取り、最初の値が2番目の値と比べて小さい、等しい、または大きい のいずれであるかを示す値-1、0、または+1を返します。ソート比較関数がnullを返す場合は、標準の組み込み 比較子が使用されます。

この**sortComparer** プロパティを使用すると、カスタム比較アルゴリズムを 提供でき、単純な文字列比較より、ユーザーが期待する結果によく 一致するソートシーケンスが得られる場合があります。

たとえば、 **Dave Koeleの英数字アルゴリズム**があります。 このアルゴリズムは、文字列を文字列や数値から成る部分に分割した後、 数値部分は値順に、文字列部分はASCII順にソートします。 Daveは、この結果を「自然なソート順」と呼んでいます。

次の例は、**sortComparer** プロパティの一般的な使用方法を示します。

```
// カスタムソート比較子を使用してCollectionViewを作成します
var dataCustomSort = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b)
      ? alphanum(a, b) // 文字列に使用するカスタム比較子
      : null; // 文字列以外にはデフォルトの比較子を使用します
  }
});
```

次の例は、 **Intl.Collator** を使用してソート順を制御する方法を示しています。

```
// Intl.Collatorを使用してソートするCollectionViewを作成します
var collator = window.Intl ? new Intl.Collator() : null;
var dataCollator = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b) && collator
      ? collator.compare(a, b) // 文字列に使用するカスタム比較子
      : null; // 文字列以外にはデフォルトの比較子を使用します
  }
});
```

継承元 **CollectionView**
型 **Function**

sortConverter

ソート時の値の変換に使用される関数を取得または設定します。

この関数は、**SortDescription**、データ項目、および変換する値をパラメーターとして受け取り、変換後の値を返す必要があります。

このプロパティはソートの動作をカスタマイズする手段を提供します。たとえば、**FlexGrid** コントロールはこのプロパティを使用して、マップされた列を生の値ではなく表示値を基準にソートします。

以下のサンプルコードは、国コードの整数を含む'country'プロパティをソートするときに、 対応する国名を使用してソートされるようにします。

```
var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(',');
collectionView.sortConverter = function (sd, item, value) {
  if (sd.property == 'countryMapped') {
    value = countries[value]; // 国IDを国名に変換します。
  }
  return value;
}
```

継承元 **CollectionView**
型 **Function**

sortDescriptions

コレクションの項目をビューでどのようにソートするかを記述する**SortDescription** オブジェクトの配列を取得します。

継承元 **CollectionView**
型 **ObservableArray**

sortNullsFirst

Gets or sets a value that determines whether null values should appear first or last when the collection is sorted (regardless of sort direction).

This property is false by default, which causes null values to appear last on the sorted collection. This is also the default behavior in Excel.

継承元 **CollectionView**
型 **boolean**

sortOnServer

ソート操作がサーバー側とクライアント側のどちらで実行されるかを決定する値を取得または設定します。

データのソート方法は**sortDescriptions** プロパティによって指定します。

The default value for this property is **true**.

型 **boolean**

sourceCollection

基になる（フィルタリングもソートもされていない）コレクションを取得または設定します。

継承元 **CollectionView**
型 **any**

tableName

このコレクションがバインドされているテーブル（エンティティ）の名前を取得します。

型 **string**

totalItemCount

ページングを適用する前のビュー内の項目の合計数を取得します。

型 **number**

trackChanges

コントロールがデータの変更を追跡するかどうかを決定する値を取得または設定します。

trackChanges がtrueに設定されている場合、**CollectionView** は、データの変更を追跡し、 **itemsAdded**、**itemsRemoved**、**itemsEdited** の各コレクションを介して変更を公開します。

変更の追跡は、変更が有効であることをユーザーが確認した後にサーバーを更新する必要がある場合に役立ちます。

変更をコミットまたはキャンセルしたら、**clearChanges** メソッドを使用して、 **itemsAdded**、**itemsRemoved**、**itemsEdited** の各コレクションをクリアします。

CollectionView は、適切な**CollectionView** メソッド（**editItem** /**commitEdit**、 **addNew** /**commitNew**、**remove**）を使用して行われた変更だけを追跡します。 データに直接加えた変更は追跡されません。

継承元 **CollectionView**
型 **boolean**

useStableSort

安定したソートアルゴリズムを使用するかどうかを取得または設定します。

安定したソートアルゴリズムは、同じキーを持つレコード間の相対的な順序を維持します。 たとえば、"Amount"フィールドを持つオブジェクトのコレクションを考えてみます。 このコレクションを"Amount"でソートする場合、安定したソートでは、Amount値が同じレコード間で元の順序が保たれます。

このプロパティのデフォルトはfalseです。この場合は、高速だが安定ではない JavaScriptの組み込みソートメソッドが**CollectionView** で使用されます。**useStableSort** をtrueに設定すると、ソート時間が30%～50%も長くなります。 コレクションが大きいと、かなりの時間の増加になります。

継承元 **CollectionView**
型 **boolean**

メソッド

addNew

```
addNew(): any
```

新しい項目を作成し、コレクションに追加します。

このメソッドは、パラメータを受け取りません。新しい項目が**commitNew** メソッドでコミットされるか、 **cancelNew** メソッドでキャンセルされるまで、 リフレッシュ操作を保留します。

次のコードは、**addNew** メソッドの典型的な使用方法を示します。

```
// 新しい項目を作成し、それをコレクションに追加します
var newItem = view.addNew();

// 新しい項目を初期化します
newItem.id = getFreshId();
newItem.name = '新しい顧客';

// ビューをリフレッシュできるように新しい項目をコミットします
view.commitNew();
```

新しい項目を**sourceCollection** にプッシュしてから、**refresh** メソッドを 呼び出すことで、新しい項目を追加することもできます。**addNew** では、 ユーザーが追加操作をキャンセルできるため、ユーザー対話式のシナリオ （データグリッドに新しい項目の追加するなど）で特に便利です。また、追加操作がコミットされるまで、新しい項目がソートされたり、 フィルタ処理でビューから除外されないようにします。

継承元 **CollectionView**
戻り値 **any**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで更新を中断します。

継承元 **CollectionView**
戻り値 **void**

cancelEdit

```
cancelEdit(): void
```

現在の編集トランザクションを終了し、可能であれば項目を元の値に戻します。

継承元 **CollectionView**
戻り値 **void**

cancelNew

```
cancelNew(): void
```

現在の追加トランザクションを終了し、追加前の新しい項目を破棄します。

継承元 **CollectionView**
戻り値 **void**

clearChanges

```
clearChanges(): void
```

itemsAdded 、**itemsRemoved** 、**itemsEdited** の各コレクションの全項目をクリアすることによってすべての変更をクリアします。

このメソッドは、変更をサーバーに確定した後またはデータをサーバーから更新した後に呼び出します。

継承元 **CollectionView**
戻り値 **void**

commitEdit

```
commitEdit(): void
```

commitEdit をオーバーライドして、データベース内の項目を変更します。

戻り値 **void**

commitNew

```
commitNew(): void
```

commitNew をオーバーライドして、データベースに新しい項目を追加します。

戻り値 **void**

contains

```
contains(item: any): boolean
```

指定した項目がこのビューに属するかどうかを示す値を返します。

パラメーター
- **item**: **any**
  調べる項目。

継承元 **CollectionView**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター
- **fn**: **Function**
  更新なしで実行する関数。

継承元 **CollectionView**
戻り値 **void**

editItem

```
editItem(item: any): void
```

指定した項目の編集トランザクションを開始します。

パラメーター

- **item**: **any**
  編集する項目。

継承元 **CollectionView**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された更新を再開します。

継承元 **CollectionView**
戻り値 **void**

getAggregate

```
getAggregate(aggType: Aggregate, binding: string, currentPage?: boolean): void
```

このコレクション内の項目の集計値を計算します。

パラメーター

- **aggType**: **Aggregate**
  計算する集計のタイプ。

- **binding**: **string**
  集計するプロパティ。

- **currentPage**: **boolean** OPTIONAL
  現在のページの項目だけを含めるかどうか。

継承元 **CollectionView**
戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **CollectionView**
戻り値 **boolean**

load

```
load(): void
```

ODataソースからデータをロードまたは再ロードします。

戻り値 **void**

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

指定した項目をビューの現在の項目に設定します。

パラメーター

- **item**: **any**
  現在の項目として設定する項目。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

ビューの最初の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

ビューの最後の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

ビューの現在の項目の後の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

ビューの指定したインデックスにある項目を現在の項目として設定します。

パラメーター

- **index**: **number**
  現在の項目として設定する項目のインデックス。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToPrevious

```
moveCurrentToPrevious(): boolean
```

ビューの現在の項目の前の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveToFirstPage

```
moveToFirstPage(): boolean
```

最初のページを現在のページとして設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveToLastPage

```
moveToLastPage(): boolean
```

最後のページを現在のページとして設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveToNextPage

```
moveToNextPage(): boolean
```

現在のページの後のページに移動します。

継承元 **CollectionView**
戻り値 **boolean**

moveToPage

```
moveToPage(index: number): boolean
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先ページのインデックス。

継承元 **CollectionView**
戻り値 **boolean**

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

現在のページの前のページに移動します。

継承元 **CollectionView**
戻り値 **boolean**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

collectionChanged イベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  変更の記述が含まれます。

継承元 **CollectionView**
戻り値 **void**

onCurrentChanged

```
onCurrentChanged(e?: EventArgs): void
```

currentChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **CollectionView**
戻り値 **void**

onCurrentChanging

```
onCurrentChanging(e: CancelEventArgs): boolean
```

currentChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs**。

継承元 **CollectionView**
戻り値 **boolean**

onError

```
onError(e: RequestErrorEventArgs): boolean
```

error イベントを発生させます。

デフォルトでは、エラーによって例外が生成され、データのリフレッシュがトリガされます。この動作を回避するには、イベントハンドラで **cancel** パラメータをtrueに設定します。

パラメーター

- **e**: **RequestErrorEventArgs**
  エラーに関する情報を含む**RequestErrorEventArgs**。

戻り値 **boolean**

onLoaded

```
onLoaded(e?: EventArgs): void
```

loaded イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLoading

```
onLoading(e?: EventArgs): void
```

loading イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onPageChanged

```
onPageChanged(e?: EventArgs): void
```

pageChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **CollectionView**
戻り値 **void**

onPageChanging

```
onPageChanging(e: PageChangingEventArgs): boolean
```

pageChanging イベントを発生させます。

パラメーター

- **e**: **PageChangingEventArgs**
  イベントデータを含む**PageChangingEventArgs**。

戻り値 **boolean**

onSourceCollectionChanged

```
onSourceCollectionChanged(e?: EventArgs): void
```

sourceCollectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **CollectionView**
戻り値 **void**

onSourceCollectionChanging

```
onSourceCollectionChanging(e: CancelEventArgs): boolean
```

sourceCollectionChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs**。

継承元 **CollectionView**
戻り値 **boolean**

refresh

```
refresh(): void
```

現在のソート、フィルタ、およびグループパラメーターを使用してビューを再作成します。

継承元 **CollectionView**
戻り値 **void**

remove

```
remove(item: any): void
```

remove をオーバーライドして、データベースから項目を削除します。

パラメーター

- **item**: **any**
  データベースから削除する項目。

戻り値 **void**

removeAt

```
removeAt(index: number): void
```

指定したインデックスにある項目をコレクションから削除します。

パラメーター

- **index**: **number**
  コレクションから削除する項目のインデックス。 このインデックスは、ソースコレクションに対してではなくビューに対する相対インデックスです。

継承元 **CollectionView**
戻り値 **void**

updateFilterDefinition

```
updateFilterDefinition(filterProvider: any): void
```

FlexGridFilter などの既知のフィルタプロバイダに基づいてフィルタ定義を更新します。

パラメーター

- **filterProvider**: **any**
  既知のフィルタプロバイダ。通常は **FlexGridFilter** の インスタンス。

戻り値 **void**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 **CollectionView**
引数 **NotifyCollectionChangedEventArgs**

currentChanged

現在の項目が変更された後に発生します。

継承元 **CollectionView**
引数 **EventArgs**

currentChanging

現在の項目が変更される前に発生します。

継承元 **CollectionView**
引数 **CancelEventArgs**

error

データの読み込みまたは書き込みエラーがあるときに発生します。

引数 **RequestErrorEventArgs**

loaded

ODataCollectionView がデータのロードを終了するときに発生します。

引数 **EventArgs**

loading

ODataCollectionView がデータのロードを開始するときに発生します。

引数 **EventArgs**

pageChanged

ページインデックスが変更された後に発生します。

継承元 **CollectionView**
引数 **EventArgs**

pageChanging

ページインデックスが変更される前に発生します。

継承元 **CollectionView**
引数 **PageChangingEventArgs**

sourceCollectionChanged

sourceCollection プロパティの値が変化した後に発生します。

継承元 **CollectionView**
引数 **EventArgs**

sourceCollectionChanging

sourceCollection プロパティの値が変化する前に発生します。

継承元	**CollectionView**
引数	**CancelEventArgs**

ODataVirtualCollectionView クラス

ファイル wijmo.odata.js
モジュール **wijmo.odata**
基本クラス **ODataCollectionView**
表示 ☐継承されたメンバー ☐イベント発生元

ODataCollectionView クラスを拡張して、 **setWindow** メソッドを使用したデータのオンデマンドロードをサポートします。

以下のサンプルコードは、**ODataCollectionView** を宣言し、 それを**FlexGrid** コントロールと同期させてグリッドの ビューポートにデータをロードする方法を示します。

```
// 仮想コレクションビューを宣言します。
var vcv = new wijmo.odata.ODataVirtualCollectionView(url, 'Order_Details_Extendeds', {
  oDataVersion: 4
});

// 仮想コレクションをグリッドのデータソースとして使用します。
flex.itemsSource = vcv;

// グリッドがスクロールしたときにデータウィンドウを更新します。
flex.scrollPositionChanged.addHandler(function () {
  var rng = flex.viewRange;
  vcv.setWindow(rng.row, rng.row2);
});
```

ODataVirtualCollectionView クラスは「データウィンドウ」を実装しており、 実際に表示されるデータのみがサーバーからロードされます。表示されていない項目はコレクションにnull値として追加され、**setWindow** メソッドの呼び出しによってロードされないかぎりnull値の ままになります。

このオンデマンドでデータをロードするメソッドは、必要になるまでデータがロードされないため、大きなデータセットを扱うときに便利です。ただし、制限もいくつかあります。ソートとフィルタリングはサーバー側で実行する必要があります。グループ化とページングはサポートされていません。

次の例では、**ODataVirtualCollectionView** を使用してNorthWind ODataプロバイダサービスからデータをロードしています。ユーザーがグリッドをスクロールするに伴ってコレクションはデータを要求時にロードします。

サンプル

Show me (https://jsfiddle.net/Wijmo5/smh5p6xr)

コンストラクタ

- constructor

プロパティ

- canAddNew
- canCancelEdit
- canChangePage
- canFilter
- canGroup
- canRemove
- canSort
- currentAddItem
- currentEditItem
- currentItem
- currentPosition
- dataTypes
- expand
- fields
- filter
- filterDefinition

- filterDefinition
- filterOnServer
- getError
- groupDescriptions
- groups
- inferDataTypes
- isAddingNew
- isEditingItem
- isEmpty
- isLoading
- isPageChanging
- isUpdating
- itemCount
- items
- itemsAdded
- itemsEdited
- itemsRemoved
- jsonReviver
- keys
- newItemCreator
- oDataVersion
- pageCount
- pageIndex
- pageOnServer
- pageSize
- requestHeaders
- showDatesAsGmt
- sortComparer
- sortConverter
- sortDescriptions
- sortNullsFirst
- sortOnServer
- sourceCollection
- tableName
- totalItemCount
- trackChanges
- useStableSort

メソッド

- addNew
- beginUpdate
- cancelEdit
- cancelNew
- clearChanges
- commitEdit
- commitNew
- contains
- deferUpdate
- editItem

- endUpdate
- getAggregate
- implementsInterface
- load
- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage
- onCollectionChanged
- onCurrentChanged
- onCurrentChanging
- onError
- onLoaded
- onLoading
- onPageChanged
- onPageChanging
- onSourceCollectionChanged
- onSourceCollectionChanging
- refresh
- remove
- removeAt
- setWindow
- updateFilterDefinition

イベント

- collectionChanged
- currentChanged
- currentChanging
- error
- loaded
- loading
- pageChanged
- pageChanging
- sourceCollectionChanged
- sourceCollectionChanging

コンストラクタ

constructor

```
constructor(url: string, tableName: string, options?: any): ODataVirtualCollectionView
```

ODataVirtualCollectionView クラスの新しいインスタンスを初期化します。

パラメーター

- **url**: **string**
  ODataサービスのURL（例: 'http://services.odata.org/Northwind/Northwind.svc'）。
- **tableName**: **string**
  サービスから取得するテーブル（エンティティ）の名前。指定されない場合は、有効なテーブル（エンティティ）のリストが取得されます。
- **options**: **any** OPTIONAL
  ODataVirtualCollectionView の初期化データ（プロパティ値とイベントハンドラ）を含むJavaScriptオブジェクト。

戻り値 **ODataVirtualCollectionView**

プロパティ

canAddNew

コレクションに新しい項目を追加できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canCancelEdit

適用前の変更を破棄して編集されたオブジェクトの元の値を復元できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canChangePage

pageIndex 値を変更できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canFilter

このビューが**filter** プロパティによってフィルタリングをサポートしているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canGroup

ODataVirtualCollectionView では、**canGroup** をfalseに設定する必要があります。

型 **boolean**

canRemove

コレクションから項目を削除できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canSort

このビューが**sortDescriptions** プロパティによってソートをサポートしているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

currentAddItem

現在の追加トランザクションの間に追加される項目を取得します。

継承元 **CollectionView**
型 **any**

currentEditItem

現在の編集トランザクションの間に編集される項目を取得します。

継承元 **CollectionView**
型 **any**

currentItem

ビューの現在の項目を取得します。

継承元 **CollectionView**
型 **any**

currentPosition

ビューの現在の項目の順序位置を取得します。

継承元 **CollectionView**
型 **number**

dataTypes

データのロード中にデータ型を変換するためのマップとして使用されるJavaScriptオブジェクトを取得または設定します。

オブジェクトのキーはフィールド名を表し、値はそのデータをどのように型変換するかを示す**DataType** 値を表します。

以下のサンプルコードは、**ODataCollectionView** を作成した後、（データベースに文字列として格納されている）'Freight'の 値を数値に変換し、さらに3つの日付フィールドを日付に変換します。

```
var orders = new wijmo.data.ODataCollectionView(url, 'Orders', {
  dataTypes: {
    Freight: wijmo.DataType.Number
    OrderDate: wijmo.DataType.Date,
    RequiredDate: wijmo.DataType.Date,
    ShippedDate: wijmo.DataType.Date,
  }
});
```

このプロパティは、データベースに格納されているデータの書式が一般的な用法に適合していない場合に役立ちます。

inferDataTypes プロパティによってDate値の変換が自動的に処理されるため、ほとんどの場合、データ型に関する情報を提供する必要はありません。

明示的に型情報を提供した場合、**inferDataTypes** プロパティは適用されません。そのため、データ型の情報を提供する場合は、Date型 のフィールドも含め、すべてのフィールドの型情報を提供する必要があります。

継承元 **ODataCollectionView**
型 **any**

expand

関連エンティティを返すデータに含めるかどうかを指定する文字列を取得または設定します。

このプロパティは直接ODataの$expandオプションにマップされます。

たとえば、次のコードは、すべての顧客とその注文をデータベースから取得します。各顧客エンティティには、注文オブジェクトの配列を含む[Orders]フィールドが含まれています。

```
var url = 'http://services.odata.org/Northwind/Northwind.svc';
var customersOrders = new wijmo.odata.ODataCollectionView(url, 'Customers', {
  expand: 'Orders'
});
```

継承元 **ODataCollectionView**
型 **string**

fields

データソースから取得するフィールドの名前を含む配列を取得または設定します。

このプロパティがnullまたは空の配列に設定されている場合は、すべてのフィールドが 取得されます。

たとえば、以下のコードは、データベースの'Categories'テーブルから3つの フィールドのみを取得する **ODataCollectionView** を作成します。

```
var categories = new wijmo.data.ODataCollectionView(url, 'Categories', {
  fields: ['CategoryID', 'CategoryName', 'Description']
});
```

継承元 **ODataCollectionView**
型 **string[]**

filter

項目がビューに含める対象として適しているかどうかを判断するために使用されるコールバックを取得または設定します。

このコールバック関数がtrueを返した場合、パラメーターとして渡された項目はビューに含まれます。

メモ: フィルタ関数でスコープ（すなわち、有効な'this'値）が必要な場合は、'this'オブジェクトを指定した'bind'関数を使用してフィルタを設定します。例:

```
collectionView.filter = this._filter.bind(this);
```

継承元 **CollectionView**
型 **IPredicate**

filterDefinition

サーバーでデータをフィルタ処理するために使用されるODataフィルタ仕様 を含む文字列を取得または設定します。

フィルタ定義構文については、 **ODataドキュメント**を参照してください。

たとえば、以下のコードでは、'CompanyName'フィールドが'A'で始まり'S'で終わる レコードがサーバーから返されます。

```
view.filterDefinition = "startswith(CompanyName, 'A') and endswith(CompanyName, 'B')";
```

フィルタ定義は自動生成することができます。たとえば、 **FlexGridFilter** コンポーネントは、データソースが **ODataCollectionView** かどうかを検出し、**filter** プロパティと **filterDefinition** プロパティの両方を自動的に更新します。

filterDefinition プロパティは、**filterOnServer** プロパティが falseに設定されている場合でも適用されることに注意してください。 これにより、同じコレクションに対してサーバーフィルタとクライアントフィルタの両方を適用でき、さまざまなシナリオで役立ちます。

たとえば、次のコードは、**filterDefinition** プロパティを使用して サーバーでフィルタ処理を実行し、 **filter** プロパティを使用して クライアントでさらにフィルタ処理を実行します。結果のコレクションには、名前が'C'で始まり、単価が20より 大きい項目が表示されます。

```
var url = 'http://services.odata.org/V4/Northwind/Northwind.svc/';
var data = new wijmo.odata.ODataCollectionView(url, 'Products', {
  oDataVersion: 4,
  filterDefinition: 'startswith(ProductName, \'C\')', // サーバー側のフィルタ
  filterOnServer: false, // クライアント側のフィルタ
  filter: function(product) {
    return product.UnitPrice > 20;
  },
});
```

継承元 **ODataCollectionView**
型 **string**

filterOnServer

ODataVirtualCollectionView では、**filterOnServer** をtrueに設定する必要があります。

型 **boolean**

getError

項目の特定のプロパティに検証エラーが含まれているかどうかを判定するコールバックを取得または設定します。

指定すると、コールバックは、検証する項目とプロパティを含む2つの パラメータを受け取り、エラーを説明する文字列を返します （エラーがない場合はnull）。

次に例を示します。

```
var view = new wijmo.collections.CollectionView(data, {
    getError: function (item, property) {
        switch (property) {
            case 'country':
                return countries.indexOf(item.country) < 0
                    ? 'Invalid Country'
                    : null;
            case 'downloads':
            case 'sales':
            case 'expenses':
                return item[property] < 0
                    ? '負にはできません。'
                    : null;
            case 'active':
                return item.active && item.country.match(/US|UK/)
                    ? 'USまたはUKではアクティブな項目が許可されません。'
                    : null;
        }
        return null;
    }
});
```

継承元 **CollectionView**
型 **Function**

groupDescriptions

コレクションの項目をビューでどのようにグループ化するかを記述する **GroupDescription** オブジェクトのコレクションを取得します。

継承元 **CollectionView**
型 **ObservableArray**

groups

最上位レベルのグループを表す **CollectionViewGroup** オブジェクトの配列を取得します。

継承元 **CollectionView**
型 **CollectionViewGroup**[]

inferDataTypes

標準の日付表現のような文字列を含むフィールドを 自動的に日付に変換するかどうかを決定する 値を取得または設定します。

ODataCollectionView クラスは、DateオブジェクトをサポートしていないJSON形式を 使用するため、このプロパティはデフォルトでtrueに設定されます。

dataTypes プロパティを使用して特定のタイプ情報を指定する場合、 このプロパティは無効になります。

継承元 **ODataCollectionView**
型 **boolean**

- isAddingNew

追加トランザクションが進行中であるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isEditingItem

編集トランザクションが進行中であるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isEmpty

このビューに項目が1つも含まれていないかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isLoading

ODataCollectionView が現在データをロードしていることを示す 値を取得します。

このプロパティを使用して、進捗状況インジケータを提供できます。

継承元 **ODataCollectionView**
型 **boolean**

- isPageChanging

ページインデックスが変更されているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

継承元 **CollectionView**
型

- itemCount

ページングを適用する前のビューの既知の項目の数を取得します。

継承元 **CollectionView**
型 **number**

- items

ビューの項目を取得します。

継承元 **CollectionView**
型 **any[]**

- itemsAdded

trackChanges が有効化されてから、コレクションに追加されたレコードを含む**ObservableArray** を取得します。

継承元 **CollectionView**
型 **ObservableArray**

- itemsEdited

trackChanges が有効化されてから、コレクションで編集されたレコードを含む**ObservableArray** を取得します。

継承元 **CollectionView**
型 **ObservableArray**

- itemsRemoved

trackChanges が有効化されてから、コレクションから削除されたレコードを含む**ObservableArray** を取得します。

継承元 **CollectionView**
型 **ObservableArray**

- jsonReviver

サーバーから返されるJSON値を解析するときに使用するカスタムのreviver関数を取得または設定します。

提供されている場合、関数は2つのパラメータ（キーおよび値）があり、解析された値（元の値と同じ値の場合があります）を返さなければなりません。

reviver関数の詳細については、 **JSON.parse メソッド**を参照してください。。

継承元 **ODataCollectionView**
型 **Function**

- keys

キーフィールドの名前を含む配列を取得または設定します。

更新操作（追加/削除/完全削除）にはキーフィールドが必要です。

継承元 **ODataCollectionView**
型 **string[]**

- newItemCreator

コレクションの新しい項目を作成する関数を取得または設定します。

作成関数が提供されない場合、**CollectionView** は、 適切な型の項目を初期化せずに作成しようとします。

作成関数が提供される場合、その関数は、パラメータを受け取らず、 コレクションに対して適切な型のオブジェクトを初期化して返す 必要があります。

継承元 **CollectionView**
型 **Function**

oDataVersion

サーバーによって使用されるODataのバージョンを取得または設定します。

現在、ODataサービスには1.0～4.0の4つのバージョンがあります。 最新サービスではバージョン4.0が使用されていますが、レガシーサービスも依然として数多く運用されています。

利用するサービスが実装しているODataのバージョンがわかっている場合は、**ODataCollectionView** を作成するときに、 **oDataVersion** プロパティを適切な値（1～4）に設定します（以下の例を参照）。

```
var url = 'http://services.odata.org/Northwind/Northwind.svc';
var categories = new wijmo.odata.ODataCollectionView(url, 'Categories', {
  oDataVersion: 1.0, // legacy OData source
  fields: ['CategoryID', 'CategoryName', 'Description'],
  sortOnServer: false
});;
```

利用するサービスが実装しているODataのバージョンがわからない場合（おそらく、 ODataエクスプローラアプリケーションを記述している場合）には、 バージョンは指定しないでください。その場合、**ODataCollectionView** この情報をサーバーから取得します。この操作には追加のリクエストが必要ですが、サービスURLあたり1回だけなので、オーバーヘッドは大きくありません。

継承元 **ODataCollectionView**
型 **number**

pageCount

総ページ数を取得します。

継承元 **ODataCollectionView**
型 **number**

pageIndex

現在のページの0から始まるインデックスを取得します。

継承元 **CollectionView**
型 **number**

pageOnServer

ODataVirtualCollectionView では、**pageOnServer** をtrueに設定する必要があります。

型 **boolean**

pageSize

1ページに表示する項目数を取得または設定します。

継承元 **ODataCollectionView**
型 **number**

requestHeaders

データを送信または要求するときに使用する要求ヘッダーを含むオブジェクトを取得または設定します。

このプロパティは、認証が必要なシナリオでよく使用されます。次に例を示します。

```
var categories = new wijmo.odata.ODataCollectionView(serviceUrl, 'Categories', {
  fields: ['Category_ID', 'Category_Name'],
  requestHeaders: { Authorization: db.token }
});
```

継承元 **ODataCollectionView**
型 **any**

showDatesAsGmt

日付を現地日ではなくGMTのように調整するかどうかを決定する値を取得または設定します。

継承元 **ODataCollectionView**
型

sortComparer

ソート時に値の比較に使用する関数を取得または設定します。

指定された場合、ソート比較関数は、パラメータとして任意の型の値を 2つ取り、最初の値が2番目の値と比べて小さい、等しい、または大きい のいずれであるかを示す値-1、0、または+1を返します。ソート比較関数がnullを返す場合は、標準の組み込み 比較子が使用されます。

この**sortComparer** プロパティを使用すると、カスタム比較アルゴリズムを 提供でき、単純な文字列比較より、ユーザーが期待する結果によく 一致するソートシーケンスが得られる場合があります。

たとえば、 **Dave Koeleの英数字アルゴリズム**があります。 このアルゴリズムは、文字列を文字列や数値から成る部分に分割した後、 数値部分は値順に、文字列部分はASCII順にソートします。 Daveは、この結果を「自然なソート順」と呼んでいます。

次の例は、**sortComparer** プロパティの一般的な使用方法を示します。

```
// カスタムソート比較子を使用してCollectionViewを作成します
var dataCustomSort = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b)
      ? alphanum(a, b) // 文字列に使用するカスタム比較子
      : null; // 文字列以外にはデフォルトの比較子を使用します
  }
});
```

次の例は、 **Intl.Collator** を使用してソート順を制御する方法を示しています。

```
// Intl.Collatorを使用してソートするCollectionViewを作成します
var collator = window.Intl ? new Intl.Collator() : null;
var dataCollator = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b) && collator
      ? collator.compare(a, b) // 文字列に使用するカスタム比較子
      : null; // 文字列以外にはデフォルトの比較子を使用します
  }
});
```

継承元 **CollectionView**
型 **Function**

sortConverter

ソート時の値の変換に使用される関数を取得または設定します。

この関数は、**SortDescription**、データ項目、および変換する値をパラメーターとして受け取り、変換後の値を返す必要があります。

このプロパティはソートの動作をカスタマイズする手段を提供します。たとえば、**FlexGrid** コントロールはこのプロパティを使用して、マップされた列を生の値ではなく表示値を基準にソートします。

以下のサンプルコードは、国コードの整数を含む'country'プロパティをソートするときに、 対応する国名を使用してソートされるようにします。

```
var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(',');
collectionView.sortConverter = function (sd, item, value) {
  if (sd.property == 'countryMapped') {
    value = countries[value]; // 国IDを国名に変換します。
  }
  return value;
}
```

継承元 **CollectionView**
型 **Function**

sortDescriptions

コレクションの項目をビューでどのようにソートするかを記述する**SortDescription** オブジェクトの配列を取得します。

継承元 **CollectionView**
型 **ObservableArray**

sortNullsFirst

Gets or sets a value that determines whether null values should appear first or last when the collection is sorted (regardless of sort direction).

This property is false by default, which causes null values to appear last on the sorted collection. This is also the default behavior in Excel.

継承元 **CollectionView**
型 **boolean**

sortOnServer

ODataVirtualCollectionView では、**sortOnServer** をtrueに設定する必要があります。

型 **boolean**

sourceCollection

基になる（フィルタリングもソートもされていない）コレクションを取得または設定します。

継承元 **CollectionView**
型 **any**

tableName

このコレクションがバインドされているテーブル（エンティティ）の名前を取得します。

継承元 **ODataCollectionView**
型 **string**

totalItemCount

ページングを適用する前のビュー内の項目の合計数を取得します。

継承元 **ODataCollectionView**
型 **number**

trackChanges

コントロールがデータの変更を追跡するかどうかを決定する値を取得または設定します。

trackChanges がtrueに設定されている場合、**CollectionView** は、データの変更を追跡し、 **itemsAdded**、**itemsRemoved**、**itemsEdited** の各コレクションを介して変更を公開します。

変更の追跡は、変更が有効であることをユーザーが確認した後にサーバーを更新する必要がある場合に役立ちます。

変更をコミットまたはキャンセルしたら、**clearChanges** メソッドを使用して、 **itemsAdded**、**itemsRemoved**、**itemsEdited** の各コレクションをクリアします。

CollectionView は、適切な**CollectionView** メソッド（**editItem** /**commitEdit**、 **addNew** /**commitNew**、**remove**）を使用して行われた変更だけを追跡します。 データに直接加えた変更は追跡されません。

継承元 **CollectionView**
型 **boolean**

useStableSort

安定したソートアルゴリズムを使用するかどうかを取得または設定します。

安定したソートアルゴリズムは、同じキーを持つレコード間の相対的な順序を維持します。 たとえば、"Amount"フィールドを持つオブジェクトのコレクションを考えてみます。 このコレクションを"Amount"でソートする場合、安定したソートでは、Amount値が同じレコード間で元の順序が保たれます。

このプロパティのデフォルトはfalseです。この場合は、高速だが安定ではない JavaScriptの組み込みソートメソッドが**CollectionView** で使用されます。**useStableSort** をtrueに設定すると、ソート時間が30%～50%も長くなります。 コレクションが大きいと、かなりの時間の増加になります。

継承元 **CollectionView**
型 **boolean**

メソッド

addNew

```
addNew(): any
```

新しい項目を作成し、コレクションに追加します。

このメソッドは、パラメータを受け取りません。新しい項目が**commitNew** メソッドでコミットされるか、 **cancelNew** メソッドでキャンセルされるまで、 リフレッシュ操作を保留します。

次のコードは、**addNew** メソッドの典型的な使用方法を示します。

```
// 新しい項目を作成し、それをコレクションに追加します
var newItem = view.addNew();

// 新しい項目を初期化します
newItem.id = getFreshId();
newItem.name = '新しい顧客';

// ビューをリフレッシュできるように新しい項目をコミットします
view.commitNew();
```

新しい項目を**sourceCollection** にプッシュしてから、**refresh** メソッドを 呼び出すことで、新しい項目を追加することもできます。**addNew** では、 ユーザーが追加操作をキャンセルできるため、ユーザー対話式のシナリオ （データグリッドに新しい項目の追加するなど）で特に便利です。また、追加操作がコミットされるまで、新しい項目がソートされたり、 フィルタ処理でビューから除外されないようにします。

継承元 **CollectionView**
戻り値 **any**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで更新を中断します。

継承元 **CollectionView**
戻り値 **void**

cancelEdit

```
cancelEdit(): void
```

現在の編集トランザクションを終了し、可能であれば項目を元の値に戻します。

継承元 **CollectionView**
戻り値 **void**

cancelNew

```
cancelNew(): void
```

現在の追加トランザクションを終了し、追加前の新しい項目を破棄します。

継承元 **CollectionView**
戻り値 **void**

clearChanges

```
clearChanges(): void
```

itemsAdded 、**itemsRemoved** 、**itemsEdited** の各コレクションの全項目をクリアすることによってすべての変更をクリアします。

このメソッドは、変更をサーバーに確定した後またはデータをサーバーから更新した後に呼び出します。

継承元 **CollectionView**
戻り値 **void**

commitEdit

```
commitEdit(): void
```

commitEdit をオーバーライドして、データベース内の項目を変更します。

継承元 **ODataCollectionView**
戻り値 **void**

commitNew

```
commitNew(): void
```

commitNew をオーバーライドして、データベースに新しい項目を追加します。

継承元 **ODataCollectionView**
戻り値 **void**

contains

```
contains(item: any): boolean
```

指定した項目がこのビューに属するかどうかを示す値を返します。

パラメーター

- **item**: **any**
  調べる項目。

継承元 **CollectionView**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

継承元 **CollectionView**
戻り値 **void**

editItem

```
editItem(item: any): void
```

指定した項目の編集トランザクションを開始します。

パラメーター

- **item**: **any**
  編集する項目。

継承元 **CollectionView**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された更新を再開します。

継承元 **CollectionView**
戻り値 **void**

getAggregate

```
getAggregate(aggType: Aggregate, binding: string, currentPage?: boolean): void
```

このコレクション内の項目の集計値を計算します。

パラメーター

- **aggType**: **Aggregate**
  計算する集計のタイプ。

- **binding**: **string**
  集計するプロパティ。

- **currentPage**: **boolean** OPTIONAL
  現在のページの項目だけを含めるかどうか。

継承元 **CollectionView**
戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **CollectionView**
戻り値 **boolean**

load

```
load(): void
```

ODataソースからデータをロードまたは再ロードします。

継承元 **ODataCollectionView**
戻り値 **void**

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

指定した項目をビューの現在の項目に設定します。

パラメーター

- **item**: **any**
  現在の項目として設定する項目。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

ビューの最初の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

ビューの最後の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

ビューの現在の項目の後の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

ビューの指定したインデックスにある項目を現在の項目として設定します。

パラメーター

- **index**: **number**
  現在の項目として設定する項目のインデックス。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToPrevious

```
moveCurrentToPrevious(): boolean
```

ビューの現在の項目の前の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveToFirstPage

```
moveToFirstPage(): boolean
```

最初のページを現在のページとして設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveToLastPage

```
moveToLastPage(): boolean
```

最後のページを現在のページとして設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveToNextPage

```
moveToNextPage(): boolean
```

現在のページの後のページに移動します。

継承元 **CollectionView**
戻り値 **boolean**

moveToPage

```
moveToPage(index: number): boolean
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先ページのインデックス。

継承元 **CollectionView**
戻り値 **boolean**

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

現在のページの前のページに移動します。

継承元 **CollectionView**
戻り値 **boolean**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

collectionChanged イベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  変更の記述が含まれます。

継承元 **CollectionView**
戻り値 **void**

onCurrentChanged

```
onCurrentChanged(e?: EventArgs): void
```

currentChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **CollectionView**
戻り値 **void**

onCurrentChanging

```
onCurrentChanging(e: CancelEventArgs): boolean
```

currentChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs**。

継承元 **CollectionView**
戻り値 **boolean**

onError

```
onError(e: RequestErrorEventArgs): boolean
```

error イベントを発生させます。

デフォルトでは、エラーによって例外が生成され、データのリフレッシュがトリガされます。この動作を回避するには、イベントハンドラで **cancel** パラメータをtrueに設定します。

パラメーター

- **e**: **RequestErrorEventArgs**
  エラーに関する情報を含む**RequestErrorEventArgs**。

継承元 **ODataCollectionView**
戻り値 **boolean**

onLoaded

```
onLoaded(e?: EventArgs): void
```

loaded イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ODataCollectionView**
戻り値 **void**

onLoading

```
onLoading(e?: EventArgs): void
```

loading イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ODataCollectionView**
戻り値 **void**

onPageChanged

```
onPageChanged(e?: EventArgs): void
```

pageChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **CollectionView**
戻り値 **void**

onPageChanging

```
onPageChanging(e: PageChangingEventArgs): boolean
```

pageChanging イベントを発生させます。

パラメーター

- **e**: **PageChangingEventArgs**
  イベントデータを含む**PageChangingEventArgs**。

継承元 **ODataCollectionView**
戻り値 **boolean**

onSourceCollectionChanged

```
onSourceCollectionChanged(e?: EventArgs): void
```

sourceCollectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **CollectionView**
戻り値 **void**

onSourceCollectionChanging

```
onSourceCollectionChanging(e: CancelEventArgs): boolean
```

sourceCollectionChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs**。

継承元 **CollectionView**
戻り値 **boolean**

refresh

```
refresh(): void
```

現在のソート、フィルタ、およびグループパラメーターを使用してビューを再作成します。

継承元 **CollectionView**
戻り値 **void**

remove

```
remove(item: any): void
```

remove をオーバーライドして、データベースから項目を削除します。

パラメーター

- **item**: **any**
  データベースから削除する項目。

継承元 **ODataCollectionView**
戻り値 **void**

removeAt

```
removeAt(index: number): void
```

指定したインデックスにある項目をコレクションから削除します。

パラメーター

- **index**: **number**
  コレクションから削除する項目のインデックス。 このインデックスは、ソースコレクションに対してではなくビューに対する相対インデックスです。

継承元 **CollectionView**
戻り値 **void**

setWindow

```
setWindow(start: number, end: number): void
```

特定のレコードの範囲をビューにロードするため、データウィンドウを設定します。

パラメーター

- **start**: **number**
  データウィンドウの最初の項目のインデックス。

- **end**: **number**
  データウィンドウの最後の項目のインデックス。

戻り値 **void**

updateFilterDefinition

```
updateFilterDefinition(filterProvider: any): void
```

FlexGridFilter などの既知のフィルタプロバイダに基づいてフィルタ定義を更新します。

パラメーター

- **filterProvider**: **any**
  既知のフィルタプロバイダ。通常は **FlexGridFilter** の インスタンス。

継承元 **ODataCollectionView**
戻り値 **void**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 **CollectionView**
引数 **NotifyCollectionChangedEventArgs**

currentChanged

現在の項目が変更された後に発生します。

継承元 **CollectionView**
引数 **EventArgs**

currentChanging

現在の項目が変更される前に発生します。

継承元 **CollectionView**
引数 **CancelEventArgs**

error

データの読み込みまたは書き込みエラーがあるときに発生します。

継承元 **ODataCollectionView**
引数 **RequestErrorEventArgs**

loaded

ODataCollectionView がデータのロードを終了するときに発生します。

継承元 **ODataCollectionView**
引数 **EventArgs**

loading

ODataCollectionView がデータのロードを開始するときに発生します。

継承元 **ODataCollectionView**
引数 **EventArgs**

pageChanged

ページインデックスが変更された後に発生します。

継承元 **CollectionView**
引数 **EventArgs**

pageChanging

ページインデックスが変更される前に発生します。

継承元 **CollectionView**
引数 **PageChangingEventArgs**

sourceCollectionChanged

sourceCollection プロパティの値が変化した後に発生します。

継承元 **CollectionView**
引数 **EventArgs**

sourceCollectionChanging

sourceCollection プロパティの値が変化する前に発生します。

継承元 **CollectionView**
引数 **CancelEventArgs**

wijmo.xlsx モジュール

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

このモジュールは**JSZip**ライブラリに依存しています。 これは、次のように参照できます。

- 同期saveメソッドおよびloadメソッドを呼び出すには、HTMLページで次のようなマークアップを使用してJSZip2ライブラリを 参照する必要があります。

```
<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js"</script>
```

- 非同期saveメソッドおよびloadメソッドを呼び出すには、HTMLページで次のようなマークアップを使用してJSZip3ライブラリを 参照する必要があります。

```
<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/3.1.3/jszip.min.js"></script>
```

- アプリケーションがnpmモジュールに基づいて構築されている場合は、スクリプトタグではなく、ES6 **import** 文を使用してJSZipモジュールをロードすることができます。この場合は、次のように、wijmo.xlsxモジュールに**useJSZip**　関数を使用してJSZipモジュールへの参照を提供する必要があります。

```
import * as JSZip from 'jszip';
import * as wjcXlsx from 'wijmo/wijmo.xlsx';
wjcXlsx.useJSZip(JSZip);
```

アプリケーションページごとに**useJSZip**　関数を1回だけ呼び出す必要があり、最良の場所は、アプリケーションのいくつかのルートモジュールです。

クラス

- DefinedName
- Workbook
- WorkbookBorder
- WorkbookBorderSetting
- WorkbookCell
- WorkbookColumn
- WorkbookFill
- WorkbookFont
- WorkbookFrozenPane
- WorkbookRow
- WorkbookStyle
- WorkbookTable
- WorkbookTableBandedStyle
- WorkbookTableBorder
- WorkbookTableColumn
- WorkbookTableCommonStyle
- WorkbookTableStyle
- WorkbookTextRun
- WorkSheet

インターフェイス

- IDefinedName
- ITableAddress
- IWorkbook
- IWorkbookBorder

- IWorkbookBorder
- IWorkbookBorderSetting
- IWorkbookCell
- IWorkbookColumn
- IWorkbookFill
- IWorkbookFont
- IWorkbookFrozenPane
- IWorkbookRow
- IWorkbookStyle
- IWorkbookTable
- IWorkbookTableBandedStyle
- IWorkbookTableBorder
- IWorkbookTableColumn
- IWorkbookTableCommonStyle
- IWorkbookTableStyle
- IWorkbookTextRun
- IWorkSheet

列挙体

- BorderStyle
- HAlign
- VAlign

メソッド

- useJSZip

メソッド

useJSZip

```
useJSZip(jszip: any): void
```

WijmoのxlsxエクスポートモジュールでJSZipモジュールへの参照を定義します。

この方法は、npmモジュールを元にしたアプリケーションでES6 import文で取得されたJSZipモジュールへの参照をwijmo.xlsxモジュールに提供するために使用されます。次に例を示します。

```
import * as JSZip from 'jszip';
import * as wjcXlsx from 'wijmo/wijmo.xlsx';
wjcXlsx.useJSZip(JSZip);
```

パラメーター

- **jszip**: **any**
  JSZipコンストラクタ関数への参照。

戻り値 **void**

DefinedName クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IDefinedName**

ワークブックオブジェクトモデルの定義された名前項目の定義を表します。

コンストラクタ

- constructor

プロパティ

- name
- sheetName
- value

コンストラクタ

constructor

```
constructor(): DefinedName
```

DefinedName クラスの新しいインスタンスを初期化します。

戻り値 **DefinedName**

プロパティ

- name

定義された名前項目の名前。

型 **string**

- sheetName

定義された名前項目がどのシートで機能するかを示します。 省略した場合、定義された名前項目はワークブックで機能します。

型 **string**

- value

定義された名前項目の値。 この値は、数式、文字列定数またはセル範囲にすることができます。 たとえば、「Sum(1, 2, 3)」、 「テスト」または「sheet1!A1:B2」。

型 **any**

Workbook クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbook**

Excelワークブックを表します。

コンストラクタ

- constructor

プロパティ

- activeWorksheet
- application
- colorThemes
- company
- created
- creator
- definedNames
- lastModifiedBy
- modified
- reservedContent
- sheets
- styles

メソッド

- fromXlsxFormat
- load
- loadAsync
- save
- saveAsync
- tableAddress
- toXlsxDateFormat
- toXlsxNumberFormat
- xlsxAddress

コンストラクタ

constructor

```
constructor(): Workbook
```

Workbook クラスの新しいインスタンスを初期化します。

戻り値 **Workbook**

プロパティ

- activeWorksheet

xlsxファイル内のアクティブなシートのインデックスを取得または設定します。

型 **number**

- application

ファイルプロパティに表示される、このファイルを生成したアプリケーションの名前を取得または設定します。

型 **string**

- colorThemes

ワークブックのテーマの色を取得します。

型 **string[]**

- company

ファイルプロパティに表示される、このファイルを生成した会社の名前を取得または設定します。

型 **string**

- created

xlsxファイルの作成時刻を取得または設定します。

型 **Date**

- creator

xlsxファイルの作成者を取得または設定します。

型 **string**

- definedNames

ワークブックで定義された名前項目を取得します。

型 **`DefinedName`[]**

- lastModifiedBy

xlsxファイルの最終変更者を取得または設定します。

型 **string**

- modified

xlsxファイルの最終変更時刻を取得または設定します。

型 **Date**

- reservedContent

flexgridまたはflexsheetがまだサポートしていないxlsxファイルの保留コンテンツを取得または設定します。

型 **any**

- sheets

ワークブックのワークシート配列を取得します。

型 **WorkSheet**[]

- styles

ワークブックのスタイルテーブルを取得します。

型 **WorkbookStyle**[]

メソッド

- STATIC fromXlsxFormat

```
fromXlsxFormat(xlsxFormat: string): string[]
```

xlsxのマルチセクション書式文字列をwijmoの対応する書式の配列に変換します。

パラメーター

- **xlsxFormat**: **string**
  Excel書式文字列。複数の書式セクションをセミコロンで区切って含めることができます。

戻り値 **string**[]

load

```
load(base64: string): void
```

base-64文字列またはデータURLからロードします。 このメソッドは、JSZip 2.5で動作します。

次に例を示します。

```
// このサンプルは、［ファイルを開く］ダイアログで選択したxlsxファイルを開き、
// ワークブックインスタンスを作成してそのファイルをロードします。


// HTML
<input type="file"
    id="importFile"
    accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
/>


// JavaScript
var workbook, // インポートされたIWorkbookを受け取ります
    importFile = document.getElementById('importFile');

importFile.addEventListener('change', function () {
    loadWorkbook();
});

function loadWorkbook() {
    var reader,
        workbook,
        file = importFile.files[0];
    if (file) {
        reader = new FileReader();
        reader.onload = function (e) {
           workbook = new wijmo.xlsx.Workbook(),
           workbook.load(reader.result);
        };
        reader.readAsDataURL(file);
    }
}
```

パラメーター

- **base64**: **string**

  xlsxファイルコンテンツを含むbase-64文字列。

戻り値 **void**

loadAsync

```
loadAsync(base64: string, onLoaded?: (workbook: Workbook), onError?: (reason?: any)): void
```

base-64文字列またはデータURLからロードします。 このメソッドは、JSZip 3.0で動作します。

パラメーター

- **base64**: **string**
  xlsxファイルコンテンツを含むbase64文字列。

- **onLoaded**: **(workbook: Workbook)** OPTIONAL
  このコールバックは、ロードされたワークブックのインスタンスを取得する方法を提供します。 このメソッドは非同期メソッドであるため、ユーザーはロードされたワークブックインスタンスを直ちに取得することができません。 このコールバックを通じてインスタンスを取得する必要があります。 コールバックのパラメータには、ロードされたワークブックのインスタンスが含まれます。 それがユーザーに渡されます。

- **onError**: **(reason?: any)** OPTIONAL
  このコールバックは、ロード時のエラー情報を捕捉します。 コールバックのパラメータには、失敗理由が含まれます。 ロード失敗の理由を確認したい場合は、この戻り値がユーザーに渡されます。

  次に例を示します。

```
workbook.loadAsync(base64, function (workbook) {

    // このコールバックで、ユーザーはロードされたワークブックインスタンスにアクセスできます。
    var app = worksheet.application ;
    ...
}, function (reason) {

    // このコールバックで、ユーザーは失敗理由を確認できます。
    console.log('The reason of load failure is ' + reason);
});
```

戻り値 **void**

save

```
save(fileName?: string): string
```

ブックをファイルに保存し、ブックのbase-64文字列表現を返します。 このメソッドは、JSZip 2.5で動作します。

たとえば、次のサンプルは、1つのセルを使用してxlsxファイルを作成します。

```
function exportXlsx(fileName) {
    var book = new wijmo.xlsx.Workbook(),
        sheet = new wijmo.xlsx.WorkSheet(),
        bookRow = new wijmo.xlsx.WorkbookRow(),
        bookCell = new wijmo.xlsx.WorkbookCell();
    bookCell.value = 'Hello, Excel!';
    bookRow.cells.push(bookCell);
    sheet.rows.push(bookRow);
    book.sheets.push(sheet);
    book.save(fileName);
}
```

ファイル名はオプションです。指定しない場合でも、メソッドは ブックを表すbase64文字列を返します。この文字列を使用して、クライアントやサーバーでその後の処理を行うことができます。

パラメーター

- **fileName**: **string** OPTIONAL
  保存するxlsxファイルの名前。

戻り値 **string**

saveAsync

```
saveAsync(fileName?: string, onSaved?: (base64?: string), onError?: (reason?: any)): void
```

非同期にブックをファイルに保存します。 このメソッドは、JSZip 3.0で動作します。

パラメーター

- **fileName**: **string** OPTIONAL
  保存するxlsxファイルの名前。

- **onSaved**: **(base64?: string)** OPTIONAL
  このコールバックは、保存されたワークブックのコンテンツを表すbase-64文字列を取得する方法を提供します。 このメソッドは非同期メソッドであるため、 ユーザーはbase-64文字列を直ちに取得することができません。 このコールバックを通じて、base-64文字列を取得する必要があります。 コールバックのパラメータには、保存されたワークブックのbase-64文字列が含まれます。 それがユーザーに渡されます。

- **onError**: **(reason?: any)** OPTIONAL
  このコールバックは、保存時のエラー情報を捕捉します。 コールバックのパラメータには、失敗理由が含まれます。 保存失敗の理由を確認したい場合は、この戻り値がユーザーに渡されます。

  次に例を示します。

```
workbook.saveAsync('', function (base64){

     // このコールバックで、ユーザーはbase64文字列にアクセスできます。
     document.getElementById('export').href = 'data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;' + 'base64
,' + base64;
}, function (reason){

     // このコールバックで、ユーザーは失敗理由を確認できます。
     console.log('The reason of save failure is ' + reason);
});
```

戻り値 **void**

STATIC tableAddress

```
tableAddress(xlsxIndex: string): ITableAddress
```

Excelの英数字のセル、行、または列のインデックスを 0 から始まる行/列インデックスのペアに変換します。

パラメーター

- **xlsxIndex**: **string**
  Aから始まる英字の列インデックス、 1 から始まる数字の行インデックス、またはその両方を含む英数字のExcelインデックス（例："D15"、"D"、"15"）。英字の列インデックスは大文字と小文字のどちらでもかまいません。 英字の列インデックスは大文字と小文字のどちらでもかまいません。

戻り値 **ITableAddress**

STATIC toXlsxDateFormat

```
toXlsxDateFormat(format: string): string
```

wijmoの日付書式をExcelの書式に変換します。

パラメーター

- **format**: **string**
  wijmoの日付書式。

戻り値 **string**

STATIC toXlsxNumberFormat

```
toXlsxNumberFormat(format: string): string
```

wijmoの数値書式をxlsxの書式に変換します。

パラメーター

- **format**: **string**
  wijmoの数値書式。

戻り値 **string**

STATIC xlsxAddress

```
xlsxAddress(row: number, col: number, absolute?: boolean, absoluteCol?: boolean): string
```

0から始まるセルの行インデックスまたは列インデックスをExcelの英数字表現に変換します。

パラメーター

- **row**: **number**
  ０から始まる行インデックス。列インデックスだけを変換する場合は、null値。
- **col**: **number**
  ０から始まる列インデックス。行インデックスだけを変換する場合は、null値。
- **absolute**: **boolean** OPTIONAL
  True値は、行インデックスと列インデックスの両方に対して絶対インデックス（例：D7）が返されることを示します。 **absoluteCol**パラメータを使用すると、この値を列インデックスに対して再定義できます。
- **absoluteCol**: **boolean** OPTIONAL
  列インデックスを絶対インデックスで返す場合、trueを指定します。

戻り値 **string**

WorkbookBorder クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
派生クラス **WorkbookTableBorder**
インターフェイス **IWorkbookBorder**

ワークブックオブジェクトモデルの境界線定義を表します。

コンストラクタ

- constructor

プロパティ

- bottom
- diagonal
- left
- right
- top

コンストラクタ

constructor

```
constructor(): WorkbookBorder
```

WorkbookBorder クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookBorder**

プロパティ

- bottom

下境界線の設定を取得または設定します。

型 **WorkbookBorderSetting**

- diagonal

斜め罫線の設定を取得または設定します。

型 **WorkbookBorderSetting**

- left

左境界線の設定を取得または設定します。

型 **WorkbookBorderSetting**

- right

右境界線の設定を取得または設定します。

型 **WorkbookBorderSetting**

- top

上境界線の設定を取得または設定します。

型 **WorkbookBorderSetting**

WorkbookBorderSetting クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookBorderSetting**

ワークブックオブジェクトモデルの背景設定定義を表します。

コンストラクタ

- constructor

プロパティ

- color
- style

コンストラクタ

constructor

```
constructor(): WorkbookBorderSetting
```

WorkbookBorderSetting クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookBorderSetting**

プロパティ

- color

境界線の設定を取得または設定します。

エクスポート操作では、任意の有効なHTML書式（6文字記法、rgb/rgba/hsl/hslaの関数形式など）で色を指定できます。 rgba/hsla表現の場合、アルファチャンネル値の指定は無視されます。

インポート操作では、色の値は常にHTMLの６文字記法（例："#afbfcf"）で表されます。

型 **string**

- style

境界線のタイプを取得または設定します。

型 **BorderStyle**

WorkbookCell クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookCell**

ワークブックオブジェクトモデルのセル定義を表します。

コンストラクタ

- constructor

プロパティ

- colSpan
- formula
- isDate
- link
- rowSpan
- style
- textRuns
- value

コンストラクタ

constructor

```
constructor(): WorkbookCell
```

WorkbookCell クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookCell**

プロパティ

- colSpan

セルのcolSpan設定を取得または設定します。

型 **number**

- formula

セルの式を取得または設定します。

型 **string**

- isDate

セルの値が日付かどうかを示します。

型 **boolean**

- link

セルのハイパーリンクを取得または設定します。

型 **string**

- rowSpan

セルのrowSpan設定を取得または設定します。

型 **number**

- style

セルのスタイルを取得または設定します。

型 **`WorkbookStyle`**

- textRuns

セルのリッチテキストを表すテキストランを取得または設定します。

型 **`WorkbookTextRun`**[]

- value

列の値を取得または設定します。

有効な値の型は、String、Number、Boolean、Dateです。

型 **any**

WorkbookColumn クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookColumn**

ワークブックオブジェクトモデルの列定義を表します。

コンストラクタ

constructor

```
constructor(): WorkbookColumn
```

WorkbookColumn クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookColumn**

プロパティ

autoWidth

セルの内容が収まるように列幅が自動的に調整されるかどうかを示す値を取得または設定します。

型 **boolean**

style

列のスタイルを取得または設定します。

このプロパティは、列内のすべてセルのスタイルを定義し、指定されたセルスタイルによってオーバーライドできます。

型 **WorkbookStyle**

visible

列の表示/非表示を取得または設定します。

型 **boolean**

width

列の幅をデバイスに依存しない（1/96インチ）ピクセル数または文字数で取得または設定します。

数値は、ピクセル単位の幅を定義します。インポート時は、幅は常にピクセル単位で表されます。

数字にサフィックス「ch」が付いた文字列値（例：'10ch'）は、幅を文字数で定義します。 これは、ExcelのUIで定義された列幅と同じ意味を持ちます。幅を文字数で指定できるのは、エクスポート操作の場合だけです。

幅が指定されていない場合は、デフォルトの幅が適用されます。

型 **any**

WorkbookFill クラス

ファイル wijmo.xlsx.js
モジュール wijmo.xlsx
インターフェイス IWorkbookFill

ワークブックオブジェクトモデルの背景塗りつぶし定義を表します。

コンストラクタ

- constructor

プロパティ

- color

コンストラクタ

constructor

```
constructor(): WorkbookFill
```

WorkbookFill クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookFill**

プロパティ

- color

塗りつぶし色を取得または設定します。

エクスポート操作では、任意の有効なHTML書式（6文字記法、rgb/rgba/hsl/hslaの関数形式など）で色を指定できます。 rgba/hsla表現の場合、アルファチャンネル値の指定は無視されます。

インポート操作では、色の値は常にHTMLの６文字記法（例："#afbfcf"）で表されます。

型 **string**

WorkbookFont クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookFont**

ワークブックオブジェクトモデルのフォント定義を表します。

コンストラクタ

- constructor

プロパティ

- bold
- color
- family
- italic
- size
- underline

コンストラクタ

constructor

```
constructor(): WorkbookFont
```

WorkbookFont クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookFont**

プロパティ

- bold

現在のフォントが太字かどうかを示します。

型 **boolean**

- color

フォントの色を取得または設定します。

エクスポート操作では、任意の有効なHTML書式（6文字記法、rgb/rgba/hsl/hslaの関数形式など）で色を指定できます。 rgba/hsla表現の場合、アルファチャンネル値の指定は無視されます。

インポート操作では、色の値は常にHTMLの６文字記法（例："#afbfcf"）で表されます。

型 **string**

- family

フォントファミリ名を取得または設定します。

型 **string**

- italic

現在のフォントに斜体スタイルが適用されているかどうかを示します。

型 **boolean**

- size

フォントサイズをデバイスに依存しない（1/96インチ）ピクセル数で取得または設定します。

型 **number**

- underline

現在のフォントが下線付きかどうかを示します。

型 **boolean**

WorkbookFrozenPane クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookFrozenPane**

ワークブックの固定ペインの定義

コンストラクタ

constructor

```
constructor(): WorkbookFrozenPane
```

WorkbookFrozenPane クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookFrozenPane**

プロパティ

columns

固定列の数を取得または設定します。

型 **number**

rows

固定行の数を取得または設定します。

型 **number**

WorkbookRow クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookRow**

ワークブックオブジェクトモデルの行定義を表します。

コンストラクタ

- constructor

プロパティ

- cells
- collapsed
- groupLevel
- height
- style
- visible

コンストラクタ

constructor

```
constructor(): WorkbookRow
```

WorkbookRow クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookRow**

プロパティ

- cells

行にあるセルの配列を取得または設定します。

行内のセルの配列を取得または設定します。配列内の各**WorkbookCell** オブジェクトは、 行内の対応する位置にあるセルを記述します。つまり、インデックス０のセルはインデックスAの列に属し、インデックス１のセルはインデックスBの列に属するセルを定義します。 xlsxファイルに特定のセルの定義が存在しない（空）場合、対応する配列要素は、 エクスポート操作とインポート操作のどちらでも未定義になります。 ある特定のセルの定義がxlsxファイルに存在しない（空である）場合、そのセルに対応する配列要素は、エクスポート操作とインポート操作のどちらにおいてもundefinedになります。

型 **WorkbookCell**[]

- collapsed

行が折りたたまれたアウトライン状態であるかどうかを示します。

型 **boolean**

- groupLevel

行のグループレベルを取得または設定します。

型 **number**

- height

行の高さをデバイスに依存しない（1/96インチ）ピクセル数で取得または設定します。

高さが指定されていない場合は、デフォルトの高さが適用されます。

型 **number**

- style

行のスタイルを取得または設定します。

このプロパティは、行内のすべてセルのスタイルを定義し、指定されたセルスタイルによってオーバーライドできます。

型 **`WorkbookStyle`**

- visible

行の表示/非表示を取得または設定します。

型 **boolean**

WorkbookStyle クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
派生クラス **WorkbookTableCommonStyle**
インターフェイス **IWorkbookStyle**

Excelのセル、列、および行のスタイル設定に使用されるワークブックオブジェクトモデルのスタイル定義を表します。

コンストラクタ

- constructor

プロパティ

- basedOn
- borders
- fill
- font
- format
- hAlign
- indent
- vAlign
- wordWrap

コンストラクタ

constructor

```
constructor(): WorkbookStyle
```

WorkbookStyle クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookStyle**

プロパティ

- basedOn

このスタイルの継承の基本スタイルを定義します。

このプロパティは、エクスポート操作にのみ適用できます。 このスタイルは、基本スタイルで定義されているすべてのプロパティを取得し、スタイル固有のプロパティを設定することで、取得したプロパティをオーバーライドまたは拡張することができます。

型 **WorkbookStyle**

- borders

境界線の設定を取得または設定します。

型 **WorkbookBorder**

- fill

背景の設定を取得または設定します。

型 **`WorkbookFill`**

- font

スタイルのフォントを取得または設定します。

型 **`WorkbookFont`**

- format

Excelの書式構文を使用して定義されたセル値の書式。

Excelの書式構文の説明については、**こちら**を参照してください。

`Workbook` クラスの**toXlsxNumberFormat**静的関数と**toXlsxDateFormat**静的関数を使用して、 .Net（**`Globalize`**）書式をExcel書式に変換できます。

型 **`string`**

- hAlign

テキストの水平方向の配置を取得または設定します。

型 **`HAlign`**

- indent

スタイルのインデント設定を取得または設定します。

型 **`number`**

- vAlign

テキストの垂直方向の配置を取得または設定します。

型 **`VAlign`**

- wordWrap

行のワードラップ設定を取得または設定します。

型 **`boolean`**

WorkbookTable クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookTable**

WorkbookTable オブジェクトモデルの背景設定定義を表します。

コンストラクタ

- constructor

プロパティ

- alterFirstColumn
- alterLastColumn
- columns
- name
- range
- showBandedColumns
- showBandedRows
- showHeaderRow
- showTotalRow
- style

コンストラクタ

constructor

```
constructor(): WorkbookTable
```

WorkbookTable クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookTable**

プロパティ

- alterFirstColumn

テーブルの最初列にスタイルが適用されるかどうかを示します。

型 **boolean**

- alterLastColumn

テーブルの最終列にスタイルが適用されるかどうかを示します。

型 **boolean**

- columns

テーブルの列。

型 **WorkbookTableColumn**[]

- name

テーブルの名前。これはプログラムによってテーブルを参照するために使用されます。

型 **string**

- range

テーブルが占める関連シート内の範囲。A1 形式の参照(「A1：D4」)で表現される。 合計行が表示されている場合、参照に含まれます。

型 **string**

- showBandedColumns

縞模様 (列)の書式設定が適用されるかどうかを示します。

型 **boolean**

- showBandedRows

縞模様 (行)の書式設定が適用されるかどうかを示します。

型 **boolean**

- showHeaderRow

テーブルにて見出し行を表示するかどうかを示します。

型 **boolean**

- showTotalRow

テーブルにて合計行を表示するかどうかを示します。

型 **boolean**

- style

テーブルで使用するテーブルスタイル。

型 **`WorkbookTableStyle`**

WorkbookTableBandedStyle クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
基本クラス **WorkbookTableCommonStyle**
インターフェイス **IWorkbookTableBandedStyle**
表示 ☐継承されたメンバー ☐イベント発生元

WorkbookTableBandedStyle オブジェクトモデルの背景設定定義を表します。

コンストラクタ

- constructor

プロパティ

- basedOn
- borders
- fill
- font
- format
- hAlign
- indent
- size
- vAlign
- wordWrap

コンストラクタ

constructor

```
constructor(): WorkbookTableBandedStyle
```

WorkbookTableBandedStyle クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookTableBandedStyle**

プロパティ

- basedOn

このスタイルの継承の基本スタイルを定義します。

このプロパティは、エクスポート操作にのみ適用できます。 このスタイルは、基本スタイルで定義されているすべてのプロパティを取得し、スタイル固有のプロパティを設定することで、取得したプロパティをオーバーライドまたは拡張することができます。

継承元 **WorkbookStyle**
型 **WorkbookStyle**

- borders

テーブルの罫線設定。

継承元 **WorkbookTableCommonStyle**
型 **WorkbookTableBorder**

- fill

背景の設定を取得または設定します。

継承元 **WorkbookStyle**
型 **WorkbookFill**

- font

スタイルのフォントを取得または設定します。

継承元 **WorkbookStyle**
型 **WorkbookFont**

- format

Excelの書式構文を使用して定義されたセル値の書式。

Excelの書式構文の説明については、**こちら**を参照してください。

Workbook クラスの**toXlsxNumberFormat**静的関数と**toXlsxDateFormat**静的関数を使用して、 .Net（**Globalize**）書式をExcel書式に変換できます。

継承元 **WorkbookStyle**
型 **string**

- hAlign

テキストの水平方向の配置を取得または設定します。

継承元 **WorkbookStyle**
型 **HAlign**

- indent

スタイルのインデント設定を取得または設定します。

継承元 **WorkbookStyle**
型 **number**

- size

単一のストライプバンド内の行数または列数。

型 **number**

- vAlign

テキストの垂直方向の配置を取得または設定します。

継承元 **WorkbookStyle**
型 **VAlign**

wordWrap

行のワードラップ設定を取得または設定します。

継承元	**WorkbookStyle**
型	**boolean**

WorkbookTableBorder クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
基本クラス **WorkbookBorder**
インターフェイス **IWorkbookTableBorder**
表示 ☐継承されたメンバー ☐イベント発生元

ワークブックオブジェクトモデルテーブルの境界線定義を表します。

コンストラクタ

- constructor

プロパティ

- bottom
- diagonal
- horizontal
- left
- right
- top
- vertical

コンストラクタ

constructor

```
constructor(): WorkbookTableBorder
```

WorkbookTableBorder クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookTableBorder**

プロパティ

- bottom

下境界線の設定を取得または設定します。

継承元 **WorkbookBorder**
型 **WorkbookBorderSetting**

- diagonal

斜め罫線の設定を取得または設定します。

継承元 **WorkbookBorder**
型 **WorkbookBorderSetting**

- horizontal

横境界線の設定。

型 **WorkbookBorderSetting**

left

左境界線の設定を取得または設定します。

継承元 **WorkbookBorder**
型 **WorkbookBorderSetting**

right

右境界線の設定を取得または設定します。

継承元 **WorkbookBorder**
型 **WorkbookBorderSetting**

top

上境界線の設定を取得または設定します。

継承元 **WorkbookBorder**
型 **WorkbookBorderSetting**

vertical

縦境界線の設定。

型 **WorkbookBorderSetting**

WorkbookTableColumn クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookTableColumn**

WorkbookTableColumn オブジェクトモデルの背景設定定義を表します。

コンストラクタ

- constructor

プロパティ

- name
- showFilterButton
- totalRowFunction
- totalRowLabel

コンストラクタ

constructor

```
constructor(): WorkbookTableColumn
```

WorkbookTableColumn クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookTableColumn**

プロパティ

- name

テーブルの列の名前。これは関数によって参照されます。

型 **string**

- showFilterButton

列に対してフィルターボタンを表示するかどうかを示します。

型 **boolean**

- totalRowFunction

列に対する合計行のセルに表示する関数。

型 **string**

- totalRowLabel

列に対する合計行のセルに表示する文字列。

型 **string**

WorkbookTableCommonStyle クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
基本クラス **WorkbookStyle**
派生クラス **WorkbookTableBandedStyle**
インターフェイス **IWorkbookTableCommonStyle**
表示 ☐継承されたメンバー ☐イベント発生元

WorkbookTableCommonStyle オブジェクトモデルの背景設定定義を表します。

コンストラクタ

- constructor

プロパティ

- basedOn
- borders
- fill
- font
- format
- hAlign
- indent
- vAlign
- wordWrap

コンストラクタ

constructor

```
constructor(): WorkbookTableCommonStyle
```

WorkbookTableCommonStyle クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookTableCommonStyle**

プロパティ

- basedOn

このスタイルの継承の基本スタイルを定義します。

このプロパティは、エクスポート操作にのみ適用できます。 このスタイルは、基本スタイルで定義されているすべてのプロパティを取得し、スタイル固有のプロパティを設定することで、取得したプロパティをオーバーライドまたは拡張することができます。

継承元 **WorkbookStyle**
型 **WorkbookStyle**

- borders

テーブルの罫線設定。

型 **WorkbookTableBorder**

- fill

背景の設定を取得または設定します。

継承元 **WorkbookStyle**
型 **WorkbookFill**

- font

スタイルのフォントを取得または設定します。

継承元 **WorkbookStyle**
型 **WorkbookFont**

- format

Excelの書式構文を使用して定義されたセル値の書式。

Excelの書式構文の説明については、**こちら**を参照してください。

Workbook クラスの**toXlsxNumberFormat**静的関数と**toXlsxDateFormat**静的関数を使用して、 .Net（**Globalize**）書式をExcel書式に変換できます。

継承元 **WorkbookStyle**
型 **string**

- hAlign

テキストの水平方向の配置を取得または設定します。

継承元 **WorkbookStyle**
型 **HAlign**

- indent

スタイルのインデント設定を取得または設定します。

継承元 **WorkbookStyle**
型 **number**

- vAlign

テキストの垂直方向の配置を取得または設定します。

継承元 **WorkbookStyle**
型 **VAlign**

- wordWrap

行のワードラップ設定を取得または設定します。

継承元 **WorkbookStyle**
型 **boolean**

WorkbookTableStyle クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookTableStyle**

WorkbookTableStyle オブジェクトモデルの背景設定定義を表します。

コンストラクタ

- constructor

プロパティ

- firstBandedColumnStyle
- firstBandedRowStyle
- firstColumnStyle
- firstHeaderCellStyle
- firstTotalCellStyle
- headerRowStyle
- lastColumnStyle
- lastHeaderCellStyle
- lastTotalCellStyle
- name
- secondBandedColumnStyle
- secondBandedRowStyle
- totalRowStyle
- wholeTableStyle

コンストラクタ

constructor

```
constructor(): WorkbookTableStyle
```

WorkbookTableStyle クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookTableStyle**

プロパティ

- firstBandedColumnStyle

「最初の列のストライプ」スタイル。

型 **WorkbookTableBandedStyle**

- firstBandedRowStyle

「最初の行のストライプ」スタイル。

型 **WorkbookTableBandedStyle**

- firstColumnStyle

「最初の列」スタイル。

型 **WorkbookTableCommonStyle**

- firstHeaderCellStyle

「最初の見出しセル」スタイル。

型 **WorkbookTableCommonStyle**

- firstTotalCellStyle

「最初の集計セル」スタイル。

型 **WorkbookTableCommonStyle**

- headerRowStyle

「見出し行」スタイル。

型 **WorkbookTableCommonStyle**

- lastColumnStyle

「最後の列」スタイル。

型 **WorkbookTableCommonStyle**

- lastHeaderCellStyle

「最後の見出しセル」スタイル。

型 **WorkbookTableCommonStyle**

- lastTotalCellStyle

「最後の集計セル」スタイル。

型 **WorkbookTableCommonStyle**

- name

テーブルスタイルの名前。

型 **string**

- secondBandedColumnStyle

「2番目の列のストライプ」スタイル。

型 **WorkbookTableBandedStyle**

- secondBandedRowStyle

「2番目の行のストライプ」スタイル。

型 **WorkbookTableBandedStyle**

- totalRowStyle

「集計行」スタイル。

型 **WorkbookTableCommonStyle**

- wholeTableStyle

「テーブル全体」スタイル。

型 **WorkbookTableCommonStyle**

WorkbookTextRun クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookTextRun**

ワークブックオブジェクトモデルのテキストラン定義を表します。

コンストラクタ

- constructor

プロパティ

- font
- text

コンストラクタ

constructor

```
constructor(): WorkbookTextRun
```

WorkbookTextRun クラスの新しいインスタンスを初期化します。

戻り値 **WorkbookTextRun**

プロパティ

- font

テキストランのフォントを取得または設定します。

型 **WorkbookFont**

- text

テキストランのテキストを取得または設定します。

型 **string**

WorkSheet クラス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkSheet**

シートのプロパティとデータを含むワークブックオブジェクトモデルのシート定義を表します。

シートのセルは行オブジェクトに格納され、**sheet.rows[i].cells[j]**のようなJavaScript式を使用してアクセスできます。

コンストラクタ

- constructor

プロパティ

- columns
- frozenPane
- name
- rows
- style
- summaryBelow
- tables
- visible

コンストラクタ

constructor

```
constructor(): WorkSheet
```

WorkSheet クラスの新しいインスタンスを初期化します。

戻り値 **WorkSheet**

プロパティ

- columns

シートの列定義の配列を取得または設定します。

列内の各**WorkbookColumn** オブジェクトは、xlsxシート内の対応する位置にある列を記述します。つまり、インデックス０の列はxlsxシートのインデックスAの列に対応し、インデックス１のオブジェクトはシートのインデックスBの列に対応します。xlsxファイルに特定の列の記述が存在しない場合、対応する配列要素は、エクスポート操作とインポート操作のどちらでも未定義になります。

配列内の**WorkbookColumn** オブジェクトで**width**プロパティの値が指定されていない場合は、デフォルトの列の幅が適用されます。

型 **WorkbookColumn**[]

- frozenPane

WorkbookFrozenPane 設定を取得または設定します。

型 **WorkbookFrozenPane**

name

シート名を取得または設定します。

型 **string**

rows

シートの行定義の配列を取得します。

配列内の各**IWorkbookRow** オブジェクトは、xlsxシート内の対応する位置にある行を記述します。つまり、インデックス０の列はxlsxシートのインデックス1の行に対応し、インデックス１のオブジェクトはシートのインデックス2の行に対応します。 xlsxファイルに特定の列のプロパティおよびデータが存在しない場合、対応する配列要素は、エクスポート操作とインポート操作のどちらでも未定義になります

配列内の**IWorkbookRow** オブジェクトで**height**プロパティの値が指定されていない場合は、デフォルトの行の高さが適用されます。

型 **WorkbookRow**[]

style

シートのスタイルを取得または設定します。

このプロパティは、ワークシート内のすべてセルのスタイルを定義し、指定されたセルスタイルによってオーバーライドできます。

型 **WorkbookStyle**

summaryBelow

サマリー行を詳細行の下に表示するか、上に表示するかを示す値を取得または設定します。

型 **boolean**

tables

このワークシートで参照されるテーブルの名前を取得します。

型 **WorkbookTable**[]

visible

ワークシートの表示/非表示を取得または設定します。

型 **boolean**

IDefinedName インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

定義された名前の定義を表します。

プロパティ

- name
- sheetName
- value

プロパティ

name

定義された名前項目の名前。

型 **string**

sheetName

定義された名前項目がどのシートで機能するかを示します。 省略した場合、定義された名前項目はワークブックで機能します。

型 **string**

value

定義された名前項目の値。 この値は、数式、文字列定数またはセル範囲にすることができます。 たとえば、「Sum(1, 2, 3)」、 「テスト」または「sheet1!A1:B2」。

型 **any**

ITableAddress インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

セルの0から始まる行インデックスおよび列インデックスと、インデックス要素が絶対的（例: "$D"）か相対的（例: "D"）かを示すプロパティを定義します。

これ以上、WorkbookTableとの関連はありません。 変換されたExcelの英数字のセルを格納する0から始まる行または列のインデックスのペアです。

プロパティ

- absCol
- absRow
- col
- row

プロパティ

- absCol

元の列インデックスが絶対インデックス（例："$D"）か相対インデックス（例："D"）かを示します。

型 **boolean**

- absRow

元の行インデックスが絶対インデックス（例：$15"）か相対インデックス（例："15"）かを示します。

型 **boolean**

- col

0から始まる列インデックス。

型 **number**

- row

0から始まる行インデックス。

型 **number**

IWorkbook インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

Excelワークブックを表します。このインタフェースは、Excelワークブックオブジェクトモデル（WOM）のルートとして、xlsxファイルに格納されるプロパティやデータを定義する方法を提供します。

xlsxファイルを作成するには、**Workbook** オブジェクトを作成し、**WorkSheet**、**WorkbookColumn**、**WorkbookRow**、**WorkbookCell** の各オブジェクトを挿入します。

xlsxファイルを保存するには、**save** メソッドを使用します。これは、ブックをファイルに保存するか、base64文字列として返すことができます。

既存のxlsxファイルをロードするには、**load** メソッドを使用してブックに挿入します。

プロパティ

- activeWorksheet
- application
- colorThemes
- company
- created
- creator
- definedNames
- lastModifiedBy
- modified
- reservedContent
- sheets
- styles

プロパティ

- activeWorksheet

xlsxファイルのアクティブシートのインデックス。

型 **number**

- application

ファイルのプロパティに表示される、このファイルを生成したアプリケーションの名前。

型 **string**

- colorThemes

ワークブックのテーマの色。

型 **string[]**

- company

ファイルのプロパティに表示される、このファイルを生成した会社の名前。

型 **string**

- created

xlsxファイルの作成日時。

型 **Date**

- creator

xlsxファイルの作成者。

型 **string**

- definedNames

定義した名前項目の配列。

型 **IDefinedName**[]

- lastModifiedBy

xlsxファイルの最終更新者。

型 **string**

- modified

xlsxファイルの最終更新日時。

型 **Date**

- reservedContent

ワークブックの保留コンテンツ。

型 **any**

- sheets

Excelワークブックのシートの配列を定義します。

型 **IWorkSheet**[]

- styles

ワークブックのスタイルテーブル。

型 **IWorkbookStyle**[]

IWorkbookBorder インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

ワークブックのセルの輪郭線の定義。

プロパティ

- bottom
- diagonal
- left
- right
- top

プロパティ

bottom

下境界線の設定。

型 **IWorkbookBorderSetting**

diagonal

斜め罫線の設定。

型 **IWorkbookBorderSetting**

left

左境界線の設定。

型 **IWorkbookBorderSetting**

right

右境界線の設定。

型 **IWorkbookBorderSetting**

top

上境界線の設定。

型 **IWorkbookBorderSetting**

IWorkbookBorderSetting インターフェイス

ファイル `wijmo.xlsx.js`
モジュール **wijmo.xlsx**

境界線のスタイル定義

プロパティ

- color
- style

プロパティ

- color

境界線の色。

エクスポート操作では、任意の有効なHTML書式（6文字記法、rgb/rgba/hsl/hslaの関数形式など）で色を指定できます。rgba/hsla表現の場合、 アルファチャンネル値の指定は無視されます。

インポート操作では、色の値は常にHTMLの６文字記法（例："#afbfcf"）で表されます。

型 **string**

- style

境界線のタイプ。

型 **`BorderStyle`**

IWorkbookCell インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

ワークブックオブジェクトモデルのセル定義を表します。

プロパティ

- colSpan
- formula
- isDate
- link
- rowSpan
- style
- textRuns
- value

プロパティ

colSpan

セルのcolSpan設定

型 **number**

formula

セルの数式

型 **string**

isDate

セルの値が日付かどうかを示します。

型 **boolean**

link

セルのハイパーリンク。

型 **string**

rowSpan

セルのrowSpan設定

型 **number**

style

セルのスタイル

型 **IWorkbookStyle**

textRuns

テキストランは、セルのリッチテキストを表します。

型 **IWorkbookTextRun**[]

value

列の値を取得または設定します。

有効な値の型は、String、Number、Boolean、Dateです。

型 **any**

IWorkbookColumn インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

ワークブックオブジェクトモデルの列定義を表します。

プロパティ

- autoWidth
- style
- visible
- width

プロパティ

- autoWidth

セルの内容が収まるように列幅が自動的に調整されるかどうかを示す値を取得または設定します。

型 **boolean**

- style

列のスタイルを取得または設定します。

このプロパティは、列内のすべてセルのスタイルを定義し、指定されたセルスタイルによってオーバーライドできます。

型 **IWorkbookStyle**

- visible

列の表示/非表示を取得または設定します。

型 **boolean**

- width

列の幅をデバイスに依存しない（1/96インチ）ピクセル数または文字数で取得または設定します。

数値は、ピクセル単位の幅を定義します。インポート時は、幅は常にピクセル単位で表されます。

数字にサフィックス「ch」が付いた文字列値（例：'10ch'）は、幅を文字数で定義します。これはExcel UIによって定義された列幅と同じ意味を持ちます。幅を文字数で指定できるのは、エクスポート操作の場合だけです。

幅が指定されていない場合は、デフォルトの幅が適用されます。

型 **any**

IWorkbookFill インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

ワークブックオブジェクトモデルの背景塗りつぶし定義を表します。

プロパティ

- color

プロパティ

- color

塗りつぶし色を取得または設定します。

エクスポート操作では、任意の有効なHTML書式（6文字記法、rgb/rgba/hsl/hslaの関数形式など）で色を指定できます。rgba/hsla表現の場合、 アルファチャンネル値の指定は無視されます。

インポート操作では、色の値は常にHTMLの６文字記法（例："#afbfcf"）で表されます。

型 **string**

IWorkbookFont インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

ワークブックオブジェクトモデルのフォント定義を表します。

プロパティ

- bold
- color
- family
- italic
- size
- underline

プロパティ

bold

このフォントが太字かどうかを示す値を取得または設定します。

型 **boolean**

color

フォントの色を取得または設定します。

エクスポート操作では、任意の有効なHTML書式（6文字記法、rgb/rgba/hsl/hslaの関数形式など）で色を指定できます。rgba/hsla表現の場合、 アルファチャンネル値の指定は無視されます。

インポート操作では、色の値は常にHTMLの６文字記法（例："#afbfcf"）で表されます。

型 **string**

family

フォントファミリ名を取得または設定します。

型 **string**

italic

このフォントに斜体スタイルが適用されるかどうかを示す値を取得または設定します。

型 **boolean**

size

フォントサイズをデバイスに依存しない（1/96インチ）ピクセル数で取得または設定します。

型 **number**

underline

このフォントが下線付きかどうかを示す値を取得または設定します。

型 **boolean**

IWorkbookFrozenPane インターフェイス

ファイル `wijmo.xlsx.js`
モジュール **wijmo.xlsx**

ワークブックの固定ペインの定義

プロパティ

- columns
- rows

プロパティ

- columns

固定列の数を取得または設定します。

型 **number**

- rows

固定行の数を取得または設定します。

型 **number**

IWorkbookRow インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

ワークブックオブジェクトモデルの行定義を表します。

プロパティ

- cells
- collapsed
- groupLevel
- height
- style
- visible

プロパティ

- cells

行にあるセルの配列を取得または設定します。

行内のセルの配列を取得または設定します。配列内の各**WorkbookCell** オブジェクトは、 行内の対応する位置にあるセルを記述します。つまり、インデックス０のセルはインデックスAの列に属し、インデックス１のセルはインデックスBの列に属するセルを定義します。ある特定のセルの定義がxlsxファイルに存在しない（空である）場合、そのセルに対応する配列要素は、エクスポート操作とインポート操作のどちらにおいてもundefinedになります。

型 **IWorkbookCell**[]

- collapsed

作成中：行が折りたたまれたアウトライン状態であるかどうかを示します。

型 **boolean**

- groupLevel

行のグループレベルを取得または設定します。

型 **number**

- height

行の高さをデバイスに依存しない（1/96インチ）ピクセル数で取得または設定します。

高さが指定されていない場合は、デフォルトの高さが適用されます。

型 **number**

- style

行のスタイルを取得または設定します。

このプロパティは、行内のすべてセルのスタイルを定義し、指定されたセルスタイルによってオーバーライドできます。

型 **IWorkbookStyle**

visible

行の表示/非表示を取得または設定します。

型 **boolean**

IWorkbookStyle インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

Excelのセル、列、および行のスタイル設定に使用されるワークブックオブジェクトモデルのスタイル定義を表します。

プロパティ

- basedOn
- borders
- fill
- font
- format
- hAlign
- indent
- vAlign
- wordWrap

プロパティ

- basedOn

このスタイルの継承の基本スタイルを定義します。

このプロパティは、エクスポート操作にのみ適用できます。 このスタイルは、基本スタイルで定義されているすべてのプロパティを取得し、スタイル固有のプロパティを設定することで、取得したプロパティをオーバーライドまたは拡張することができます。

型 **IWorkbookStyle**

- borders

セルの輪郭線の設定。

型 **IWorkbookBorder**

- fill

セルの背景。

型 **IWorkbookFill**

- font

フォントプロパティを取得または設定します。

型 **IWorkbookFont**

- format

Excelの書式構文を使用して定義されたセル値の書式。

Excelの書式構文の説明については、**こちら**を参照してください。

`Workbook` クラスの**toXlsxNumberFormat**静的関数と**toXlsxDateFormat**静的関数を使用して、 .Net（**`Globalize`**）書式をExcel書式に変換できます。

型 **string**

- hAlign

テキストの水平方向の配置を取得または設定します。

型 **`HAlign`**

- indent

テキストのインデント。 増分値 1 が 3 つのスペースを表す整数値です。

型 **number**

- vAlign

テキストの垂直方向の配置を取得または設定します。

型 **`VAlign`**

- wordWrap

ワードラップ設定。

型 **boolean**

IWorkbookTable インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

テーブル定義を表します。

プロパティ

- alterFirstColumn
- alterLastColumn
- columns
- name
- range
- showBandedColumns
- showBandedRows
- showHeaderRow
- showTotalRow
- style

プロパティ

alterFirstColumn

テーブルの最初列にスタイルが適用されるかどうかを示します。

型 **boolean**

alterLastColumn

テーブルの最終列にスタイルが適用されるかどうかを示します。

型 **boolean**

columns

テーブルの列。

型 **IWorkbookTableColumn**[]

name

テーブルの名前。これはプログラムによってテーブルを参照するために使用されます。

型 **string**

range

テーブルが占める関連シート内の範囲。A1 形式の参照(「A1：D4」)で表現される。 合計行が表示されている場合、参照に含まれます。

型 **string**

- showBandedColumns

縞模様 (列)の書式設定が適用されるかどうかを示します。

型 **boolean**

- showBandedRows

縞模様 (行)の書式設定が適用されるかどうかを示します。

型 **boolean**

- showHeaderRow

テーブルにて見出し行を表示するかどうかを示します。

型 **boolean**

- showTotalRow

テーブルにて合計行を表示するかどうかを示します。

型 **boolean**

- style

テーブルで使用するテーブルスタイル。

型 **`IWorkbookTableStyle`**

IWorkbookTableBandedStyle インターフェイス

ファイル `wijmo.xlsx.js`
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookTableCommonStyle**

テーブルのストライプスタイルの定義を表します。

プロパティ

- size

プロパティ

size

単一のストライプバンド内の行数または列数。

型 **number**

IWorkbookTableBorder インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookBorder**

テーブルの罫線定義。

プロパティ

- horizontal
- vertical

プロパティ

- horizontal

横境界線の設定。

型 **IWorkbookBorderSetting**

- vertical

縦境界線の設定。

型 **IWorkbookBorderSetting**

IWorkbookTableColumn インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

テーブル列の定義を表します。

プロパティ

- name
- showFilterButton
- totalRowFunction
- totalRowLabel

プロパティ

name

テーブルの列の名前。これは関数によって参照されます。

型 **string**

showFilterButton

列に対してフィルターボタンを表示するかどうかを示します。

型 **boolean**

totalRowFunction

列に対する合計行のセルに表示する関数。

型 **string**

totalRowLabel

列に対する合計行のセルに表示する文字列。

型 **string**

IWorkbookTableCommonStyle インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**
インターフェイス **IWorkbookStyle**

テーブルの一般的なスタイルの定義を表します。

プロパティ

- borders

プロパティ

borders

テーブルの罫線設定。

型 **IWorkbookTableBorder**

IWorkbookTableStyle インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

テーブルスタイルの定義を表します。

プロパティ

- firstBandedColumnStyle
- firstBandedRowStyle
- firstColumnStyle
- firstHeaderCellStyle
- firstTotalCellStyle
- headerRowStyle
- lastColumnStyle
- lastHeaderCellStyle
- lastTotalCellStyle
- name
- secondBandedColumnStyle
- secondBandedRowStyle
- totalRowStyle
- wholeTableStyle

プロパティ

firstBandedColumnStyle

「最初の列のストライプ」スタイル。

型 **IWorkbookTableBandedStyle**

firstBandedRowStyle

「最初の行のストライプ」スタイル。

型 **IWorkbookTableBandedStyle**

firstColumnStyle

「最初の列」スタイル。

型 **IWorkbookTableCommonStyle**

firstHeaderCellStyle

「最初の見出しセル」スタイル。

型 **IWorkbookTableCommonStyle**

- firstTotalCellStyle

「最初の集計セル」スタイル。

型 **IWorkbookTableCommonStyle**

- headerRowStyle

「見出し行」スタイル。

型 **IWorkbookTableCommonStyle**

- lastColumnStyle

「最後の列」スタイル。

型 **IWorkbookTableCommonStyle**

- lastHeaderCellStyle

「最後の見出しセル」スタイル。

型 **IWorkbookTableCommonStyle**

- lastTotalCellStyle

「最後の集計セル」スタイル。

型 **IWorkbookTableCommonStyle**

- name

テーブルスタイルの名前。

型 **string**

- secondBandedColumnStyle

「2番目の列のストライプ」スタイル。

型 **IWorkbookTableBandedStyle**

- secondBandedRowStyle

「2番目の行のストライプ」スタイル。

型 **IWorkbookTableBandedStyle**

- totalRowStyle

「集計行」スタイル。

型 **IWorkbookTableCommonStyle**

wholeTableStyle

「テーブル全体」スタイル。

型 **IWorkbookTableCommonStyle**

IWorkbookTextRun インターフェイス

ファイル `wijmo.xlsx.js`
モジュール **wijmo.xlsx**

リッチテキスト用のテキストランの部分。

プロパティ

- font
- text

プロパティ

- font

テキストランのフォント。

型 **`IWorkbookFont`**

- text

テキストランのテキスト。

型 **string**

IWorkSheet インターフェイス

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

シートのプロパティとデータを含むワークブックオブジェクトモデルのシート定義を表します。

シートのセルは行オブジェクトに格納され、**sheet.rows[i].cells[j]**のようなJavaScript式を使用してアクセスできます。

プロパティ

- columns
- frozenPane
- name
- rows
- style
- summaryBelow
- tables
- visible

プロパティ

- columns

シートの列定義の配列を取得または設定します。

列内の各**WorkbookColumn** オブジェクトは、xlsxシート内の対応する位置にある列を記述します。つまり、インデックス0の列はxlsxシートのインデックスAの列に対応し、インデックス1のオブジェクトはシートのインデックスBの列に対応します。xlsxファイルに特定の列の記述が存在しない場合、対応する配列要素は、エクスポート操作とインポート操作のどちらでも未定義になります。

配列内の**IWorkbookColumn** オブジェクトで**width**プロパティの値が指定されていない場合は、デフォルトの列幅が適用されます。

型 **IWorkbookColumn**[]

- frozenPane

固定ペインの設定を取得または設定します。

型 **IWorkbookFrozenPane**

- name

シート名を取得または設定します。

型 **string**

- rows

シートの行定義の配列を取得または設定します。

配列内の各**IWorkbookRow** オブジェクトは、xlsxシート内の対応する位置にある行を記述します。つまり、インデックス０の列はxlsxシートのインデックス1の行に対応し、インデックス１のオブジェクトはシートのインデックス2の行に対応します。xlsxファイルに特定の行の記述が存在しない場合、対応する配列要素は、エクスポート操作とインポート操作のどちらでも未定義になります。

配列内の**IWorkbookRow** オブジェクトで**height** プロパティの値が指定されていない場合は、デフォルトの行の高さが適用されます。

型 **IWorkbookRow**[]

- style

シートのスタイルを取得または設定します。

このプロパティは、ワークシート内のすべてセルのスタイルを定義し、指定されたセルスタイルによってオーバーライドできます。

型 **IWorkbookStyle**

- summaryBelow

サマリー行を詳細行の下に表示するか、上に表示するかを示す値を取得または設定します。

型 **boolean**

- tables

このワークシートでのテーブルを取得します。

型 **IWorkbookTable**[]

- visible

ワークシートの表示/非表示を取得または設定します。

型 **boolean**

BorderStyle 列挙体

ファイル `wijmo.xlsx.js`
モジュール **wijmo.xlsx**

境界線の線スタイル

メンバー

名前	値	説明
None	0	境界線なし
Thin	1	細い境界線
Medium	2	中細の境界線
Dashed	3	破線の境界線
Dotted	4	点線の境界線
Thick	5	太い境界線
Double	6	二重線の境界線
Hair	7	極細の境界線
MediumDashed	8	中細破線の境界線
ThinDashDotted	9	細1点鎖線の境界線
MediumDashDotted	10	中細2点鎖線の境界線
ThinDashDotDotted	11	細2点鎖線の境界線
MediumDashDotDotted	12	中細2点鎖線の境界線
SlantedMediumDashDotted	13	中細1点鎖斜線の境界線

HAlign 列挙体

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

ワークブックオブジェクトモデルの水平方向のテキストの配置を定義します。

メンバー

名前	値	説明
General	0	配置はセル値の型によって決まります。
Left	1	テキストは左に揃えられます。
Center	2	テキストは中央に揃えられます。
Right	3	テキストは右に揃えられます。
Fill	4	セル幅全体を満たすようにテキストが繰り返されます。
Justify	5	テキストは両端揃えで配置されます。

VAlign 列挙体

ファイル wijmo.xlsx.js
モジュール **wijmo.xlsx**

垂直方向の配置

メンバー

名前	値	説明
Top	0	垂直方向の上揃
Center	1	垂直方向の中央揃え
Bottom	2	垂直方向の下揃え
Justify	3	垂直方向の両端揃え

wijmo.pdf モジュール

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

PdfDocument クラスおよび関連クラスを定義します。

クラス

- PdfBrush
- PdfDashPattern
- PdfDocument
- PdfDocumentEndedEventArgs
- PdfFont
- PdfGradientBrush
- PdfGradientStop
- PdfLinearGradientBrush
- PdfPageArea
- PdfPaths
- PdfPen
- PdfRadialGradientBrush
- PdfRunningTitle
- PdfRunningTitleDeclarativeContent
- PdfSolidBrush

インターフェイス

- IPdfBufferedPageRange
- IPdfDocumentInfo
- IPdfFontAttributes
- IPdfFontFile
- IPdfImage
- IPdfImageDrawSettings
- IPdfPageMargins
- IPdfPageSettings
- IPdfSvgDrawSettings
- IPdfTextDrawSettings
- IPdfTextMeasurementInfo
- IPdfTextMeasurementSettings
- IPdfTextSettings

列挙体

- PdfFillRule
- PdfImageHorizontalAlign
- PdfImageVerticalAlign
- PdfLineCapStyle
- PdfLineJoinStyle
- PdfPageOrientation
- PdfPageSize
- PdfTextHorizontalAlign

メソッド

- ptToPx
- pxToPt
- saveBlob

メソッド

ptToPx

```
ptToPx(value: number): number
```

ポイント単位値をピクセル単位値に変換します。

パラメーター

- **value**: **number**
  変換する値。

戻り値 **number**

pxToPt

```
pxToPt(value: number): number
```

ピクセル単位値をポイント単位値に変換します。

パラメーター

- **value**: **number**
  変換する値。

戻り値 **number**

saveBlob

```
saveBlob(blob: Blob, fileName: string): void
```

Blobオブジェクトをファイルとして保存します。

パラメーター

- **blob**: **Blob**
  保存するBlobオブジェクト。

- **fileName**: **string**
  ファイルが保存される名前。

戻り値 **void**

PdfBrush クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
派生クラス **PdfGradientBrush**, **PdfSolidBrush**

すべてのブラシの基本クラスになる抽象クラスを表します。 このクラスから派生されるクラスのインスタンスは、領域およびテキストの塗りつぶしに使用されます。

このクラスは、ユーザーコード内でインスタンス化することを意図していません。

メソッド

メソッド

- clone

```
clone(): PdfBrush
```

この**PdfBrush** のコピーを作成します。

戻り値 **PdfBrush**

- equals

```
equals(value: PdfBrush): boolean
```

指定された**PdfBrush** インスタンスが現在のインスタンスと等しいかどうかを判定します。

パラメーター
- **value**: **PdfBrush**
  比較する**PdfBrush**。

戻り値 **boolean**

PdfDashPattern クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

パスのストロークに使用される破線パターンを表します。

コンストラクタ

- constructor

プロパティ

- dash
- gap
- phase

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(dash?: number, gap?: number, phase?: number): PdfDashPattern
```

PdfDashPattern クラスの新しいインスタンスを初期化します。

パラメーター

- **dash**: **number** OPTIONAL
  破線の長さ（ポイント単位）。
- **gap**: **number** OPTIONAL
  間隔の長さ（ポイント単位）。
- **phase**: **number** OPTIONAL
  破線パターンの破線を開始する距離（ポイント単位）。

戻り値 **PdfDashPattern**

プロパティ

- dash

破線の長さ（ポイント単位）を取得または設定します。 デフォルト値はnullです。これは破線パターンではなく、実線を示します。

型 **number**

- gap

間隔の長さ（ポイント単位）を取得または設定します。 デフォルト値は**dash** と同じで、破線と間隔が 同じ長さになります。

型 **number**

phase

破線パターンの破線を開始する距離（ポイント単位）を取得または設定します。 デフォルト値は0です。

型 **number**

メソッド

clone

```
clone(): PdfDashPattern
```

この**PdfDashPattern** のコピーを作成します。

戻り値 **PdfDashPattern**

equals

```
equals(value: PdfDashPattern): boolean
```

指定された**PdfDashPattern** インスタンスが現在のインスタンスと 等しいかどうかを判定します。

パラメーター

- **value**: **PdfDashPattern**
  比較する**PdfDashPattern**。

戻り値 **boolean**

PdfDocument クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
基本クラス **PdfPageArea**
表示 ☐継承されたメンバー ☐イベント発生元

PDFKit JavaScriptライブラリに基づくPDFドキュメントオブジェクトを表します。

コンストラクタ

- constructor

プロパティ

- bufferPages
- compress
- currentPageSettings
- document
- footer
- header
- height
- info
- lineGap
- pageIndex
- pageSettings
- paths
- width
- x
- y

メソッド

- addPage
- bufferedPageRange
- dispose
- drawImage
- drawSvg
- drawText
- end
- lineHeight
- measureText
- moveDown
- moveUp
- onEnded
- onPageAdded
- openImage
- registerFont
- registerFontAsync
- restoreState
- rotate
- saveState
- scale

- setBrush
- setFont
- setPen
- transform
- translate

イベント

- ended
- pageAdded

コンストラクタ

constructor

```
constructor(options?: any): PdfDocument
```

PdfDocument クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  初期化設定を含むオプションのオブジェクト。

戻り値 **PdfDocument**

プロパティ

- bufferPages

ページのバッファリングモードが有効かどうかを示す値を取得します。 このモードでは、**pageIndex** および**bufferedPageRange** を使用してドキュメントのページを反復処理できます。

このプロパティは、**PdfDocument** コンストラクタを使用してのみ割り当てることができます。 このプロパティは、**header** と**footer** が両方とも非表示である場合にのみfalseに設定できます。

デフォルト値はtrueです。

型 **boolean**

- compress

ドキュメント圧縮が有効かどうかを示す値を取得します。 このプロパティは、**PdfDocument** コンストラクタを使用してのみ割り当てることができます。

デフォルト値はtrueです。

型 **boolean**

- currentPageSettings

現在のページ設定を表すオブジェクトを取得します（読み取り専用）。

型 **IPdfPageSettings**

- document

ドキュメントオブジェクトを取得します。

継承元 **PdfPageArea**
型 **PdfDocument**

- footer

下マージンの直上に配置されているページ領域のフッターを表すオブジェクトを取得します。

型 **PdfRunningTitle**

- header

上マージンの直下に配置されているページ領域のヘッダーを表すオブジェクトを取得します。

型 **PdfRunningTitle**

- height

領域の高さ（ポイント単位）を取得します。

継承元 **PdfPageArea**
型 **number**

- info

作成者名やドキュメントの作成日などのドキュメント情報を取得または設定します。

型 **IPdfDocumentInfo**

- lineGap

テキスト行の間隔（ポイント単位）を取得または設定します。

デフォルト値は0です。

継承元 **PdfPageArea**
型 **number**

- pageIndex

バッファページ範囲内の現在のページのインデックスを取得または設定します。

bufferedPageRange メソッドを使用して、バッファページの範囲を取得します。

型 **number**

- pageSettings

自動的に追加されるページおよび**addPage** メソッドの デフォルトのページ設定を表すオブジェクトを取得します。

型 **IPdfPageSettings**

paths

パスを描画する機能を提供するオブジェクトを取得します。

継承元 **PdfPageArea**
型 **PdfPaths**

width

領域の幅（ポイント単位）を取得します。

継承元 **PdfPageArea**
型 **number**

x

テキストまたは画像の描画に使用されるテキストフロー内の現在の点のX座標 （ポイント単位）を取得または設定します。

継承元 **PdfPageArea**
型 **number**

y

テキストまたは画像の描画に使用されるテキストフロー内の現在の点のY座標 （ポイント単位）を取得または設定します。

継承元 **PdfPageArea**
型 **number**

メソッド

addPage

```
addPage(settings?: IPdfPageSettings): PdfDocument
```

指定された設定で新しいページを追加します。

設定パラメータを省略すると、代わりに**pageSettings** が使用されます。

パラメーター

- **settings**: **IPdfPageSettings** OPTIONAL
  ページ設定。

戻り値 **PdfDocument**

bufferedPageRange

```
bufferedPageRange(): IPdfBufferedPageRange
```

バッファページの範囲を取得します。

戻り値 **IPdfBufferedPageRange**

dispose

```
dispose(): void
```

ドキュメントを破棄します。

戻り値 **void**

drawImage

```
drawImage(src: any, x?: number, y?: number, options?: IPdfImageDrawSettings): PdfPageArea
```

指定されたオプションを使用して、JPGまたはPNG形式で画像を描画します。

xおよびyが定義されていない場合は、代わりに@see:x および@see:y が使用されます。

最後に、画像がテキストフロー内で描画された場合、このメソッドは@see:y を更新します。 したがって、後続のテキストまたは画像は、この点の下から開始されます。

パラメーター

- **src**: **any**
  画像の取得元のURLを含む文字列、Base64エンコード画像を含むデータURI、または**IPdfImage** オブジェクト。
- **x**: **number** OPTIONAL
  画像を描画するポイントのx座標（ポイント単位）。
- **y**: **number** OPTIONAL
  画像を描画するポイントのy座標（ポイント単位）。
- **options**: **IPdfImageDrawSettings** OPTIONAL
  画像描画オプションを決定します。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

drawSvg

```
drawSvg(url: string, x?: number, y?: number, options?: IPdfSvgDrawSettings): PdfPageArea
```

指定されたオプションを使用してSVG画像を描画します。

xおよびyが定義されていない場合は、代わりに@see:x および@see:y が使用されます。

このメソッドは、最も外側のSVG要素のwidth属性とheight属性の値を使用し、 options.widthプロパティとoptions.heightプロパティに基づいて拡大縮小率を決定します。これらの属性のいずれかを省略した場合、 拡大縮小は行われず、画像は元のサイズでレンダリングされます。

最後に、画像がテキストフロー内で描画された場合、このメソッドは@see:y を更新します。 したがって、後続のテキストまたは画像は、この点の下から開始されます。 インクリメント値は、options.heightプロパティまたは最も外側のSVG要素のheight属性によって定義されます。 どちらも指定されていない場合、@see:y は変更されません。

このメソッドは、SVG機能の一部のみをサポートしており、主にWijmo 5のチャートコントロールをレンダリングするために提供されています。

パラメーター

- **url**: **string**
  SVG画像の取得元のURLを含む文字列、またはBase64エンコードSVG画像を含むデータURI。
- **x**: **number** OPTIONAL
  画像を描画するポイントのx座標（ポイント単位）。
- **y**: **number** OPTIONAL
  画像を描画するポイントのy座標（ポイント単位）。
- **options**: **IPdfSvgDrawSettings** OPTIONAL
  SVG画像描画オプションを決定します。

継承元 `PdfPageArea`
戻り値 `PdfPageArea`

drawText

```
drawText(text: string, x?: number, y?: number, options?: IPdfTextDrawSettings): IPdfTextMeasurementInfo
```

指定されたオプションを使用して文字列を描画し、測定情報を返します。

options.pen、**options.brush**、または**options.font**を省略すると、 それぞれ現在のドキュメントのペン、ブラシ、またはフォントが使用されます（**setPen**、 **setBrush**、および**setFont** を参照）。

文字列は、左上隅がxおよびy、幅および高さが **options.width**値および**options.height**値で定義される四角形領域内に描画されます。xとyを指定しない場合は、代わりに**x** プロパティと**y** プロパティが使用されます。

領域内のテキストは、自動的に折り返されてクリッピングされます。 **options.height**が定義されておらず、テキストが本文の下端を超えた場合、 テキストに合わせて新しいページが追加されます。

最後に、**x** プロパティと**y** プロパティの値を更新します。したがって、後続のテキストまたは画像は、この点の下から開始されます（**options.continued**の値によって異なる）。

測定結果には、テキストが複数のページまたは列に分割される可能性があることが反映されません。 テキストは単一ブロックとして処理されます。

パラメーター

- **text**: **string**
  描画するテキスト。
- **x**: **number** OPTIONAL
  テキストを描画するポイントのX座標（ポイント単位）。
- **y**: **number** OPTIONAL
  テキストを描画するポイントのY座標（ポイント単位）。
- **options**: **IPdfTextDrawSettings** OPTIONAL
  テキスト描画オプションを決定します。

継承元 **PdfPageArea**
戻り値 **IPdfTextMeasurementInfo**

end

```
end(): void
```

ドキュメントのレンダリングを終了します。

戻り値 **void**

lineHeight

```
lineHeight(font?: PdfFont): number
```

指定されたフォントの行の高さを取得します。

フォントが指定されていない場合は、現在のドキュメントで使用されているフォントが使用されます。

パラメーター

- **font**: **PdfFont** OPTIONAL
  行の高さを取得するフォント。

継承元 **PdfPageArea**
戻り値 **number**

measureText

```
measureText(text: string, font?: PdfFont, options?: IPdfTextMeasurementSettings): IPdfTextMeasurementInfo
```

指定されたフォントおよびテキスト描画オプションを使用して、テキストをレンダリングせずに測定します。

フォントが指定されていない場合は、現在のドキュメントで使用されているフォントが使用されます。

このメソッドは、**drawText** と同じテキストレンダリングエンジンを使用するため、 options.widthが指定されていない場合は、同様に@see:x およびページの右マージンに制約されます。測定結果には、テキストが複数のページまたは列に分割される可能性があることが反映されません。 テキストは単一ブロックとして処理されます。

パラメーター

- **text**: **string**
  測定するテキスト。
- **font**: **PdfFont** OPTIONAL
  テキストに適用されるフォント。
- **options**: **IPdfTextMeasurementSettings** OPTIONAL
  テキスト描画オプションを決定します。

継承元 **PdfPageArea**
戻り値 **IPdfTextMeasurementInfo**

moveDown

```
moveDown(lines?: number, font?: PdfFont): PdfPageArea
```

指定されたフォントまたはフォントが指定されていない場合は現在のドキュメントのフォントを使用して、指定された行数だけ@see:y を下に移動します。

パラメーター

- **lines**: **number** OPTIONAL
  下に移動する行数。
- **font**: **PdfFont** OPTIONAL
  行の高さを計算するフォント。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

moveUp

```
moveUp(lines?: number, font?: PdfFont): PdfPageArea
```

指定されたフォントまたはフォントが指定されていない場合は現在のドキュメントのフォントを使用して、指定された行数だけ@see:y を上に移動します。

パラメーター

- **lines**: **number** OPTIONAL
  上に移動する行数。
- **font**: **PdfFont** OPTIONAL
  行の高さを計算するフォント。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

onEnded

```
onEnded(args: PdfDocumentEndedEventArgs): void
```

end イベントを発生させます。

パラメーター

- **args**: **PdfDocumentEndedEventArgs**
  イベントデータを含む**PdfDocumentEndedEventArgs** オブジェクト。

戻り値 **void**

onPageAdded

```
onPageAdded(args: EventArgs): void
```

pageAdded イベントを発生させます。

パラメーター

- **args**: **EventArgs**
  イベントデータを含む**EventArgs** オブジェクト。

戻り値 **void**

openImage

```
openImage(url: string): IPdfImage
```

JPGまたはPNG形式で画像を開きます。

パラメーター

- **url**: **string**
  画像の取得元のURLを含む文字列、またはBase64エンコード画像を含むデータURI。

継承元 **PdfPageArea**
戻り値 **IPdfImage**

registerFont

```
registerFont(font: IPdfFontFile): PdfDocument
```

ソースからフォントを登録し、それを指定されたフォントファミリ名およびフォント属性に関連付けます。

パラメーター

- **font**: **IPdfFontFile**
  登録するフォント。

戻り値 **PdfDocument**

registerFontAsync

```
registerFontAsync(font: IPdfFontFile, callback: Function): void
```

URLからフォントを非同期に登録し、それを指定されたフォントファミリ名 およびフォント属性に関連付けます。

このコールバック関数は、パラメータとして **IPdfFontFile** オブジェクトを受け取ります。

パラメーター

- **font**: **IPdfFontFile**
  登録するフォント。

- **callback**: **Function**
  フォントが登録されると呼び出されるコールバック関数。

戻り値 **void**

restoreState

```
restoreState(): PdfDocument
```

スタックから状態を復元し、それをグラフィックコンテキストに適用します。

戻り値 **PdfDocument**

rotate

```
rotate(angle: number, origin?: Point): PdfPageArea
```

指定された角度でグラフィックコンテキストを時計回りに回転します。

パラメーター

- **angle**: **number**
  回転角度（度単位）。

- **origin**: **Point** OPTIONAL
  回転の中心の**Point** （ポイント単位）。 指定しない場合は、左上隅が使用されます。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

saveState

```
saveState(): PdfDocument
```

グラフィックコンテキストの状態（現在のペン、ブラシ、変換状態を含む）を 保存し、それをスタックにプッシュします。

戻り値 **PdfDocument**

scale

```
scale(xFactor: number, yFactor?: number, origin?: Point): PdfPageArea
```

指定された拡大率でグラフィックコンテキストを拡大縮小します。

範囲[0, 1]内の拡大率の値は、サイズが小さくなることを示します。 1 より大きい拡大率の値は、サイズが大きくなることを示します。

パラメーター

- **xFactor**: **number**
  Xサイズを拡大縮小する係数。
- **yFactor**: **number** OPTIONAL
  Yサイズを拡大縮小する係数。 指定しない場合は、xFactorと等しい値と見なされます。
- **origin**: **Point** OPTIONAL
  拡大縮小の中心となる**Point** （ポイント単位）。 指定しない場合は、左上隅が使用されます。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

setBrush

```
setBrush(brushOrColor: any): PdfDocument
```

デフォルトのドキュメントブラシを設定します。 特定のブラシが提供されていない場合は、このブラシが **fill**、 **fillAndStroke**、および**drawText** メソッドで使用されます。

brushOrColor引数は、次の値を取ることができます。

- **PdfBrush** オブジェクト。
- **Color** オブジェクト、または**fromString** メソッドが受け取ることができる任意の文字列。 この場合、指定された色の**PdfBrush** オブジェクトが内部的に作成されます。

パラメーター

- **brushOrColor**: **any**
  使用するブラシまたは色。

戻り値 **PdfDocument**

setFont

```
setFont(font: PdfFont): PdfDocument
```

ドキュメントフォントを設定します。 指定されたスタイルおよび重みプロパティを持つフォントが見つからない場合は

- **フォント重みのフォールバック** を使用して、 最も近いフォントが検索されます。
- 何も見つからなかった場合は、以下の順序で他のスタイルの最も近いフォントが検索されます。
  - **'italic'**: 'oblique', 'normal'.
  - **'oblique'**: 'italic', 'normal'.
  - **'normal'**: 'oblique', 'italic'.

パラメーター

- **font**: **PdfFont**
  設定するフォントオブジェクト。

戻り値 **PdfDocument**

setPen

```
setPen(penOrColor: any): PdfDocument
```

デフォルトのドキュメントペンを設定します。 特定のペンが提供されていない場合は、このペンが**stroke** 、 **fillAndStroke** 、および**drawText** メソッドで使用されます。

penOrColor引数は、次の値を取ることができます。

- **PdfPen** オブジェクト。
- **Color** オブジェクト、または**fromString** メソッドが受け取ることができる任意の文字列。 この場合、指定された色の**PdfPen** オブジェクトが内部的に作成されます。

パラメーター

- **penOrColor**: **any**
  使用するペンまたは色。

戻り値 **PdfDocument**

transform

```
transform(a: number, b: number, c: number, d: number, e: number, f: number): PdfPageArea
```

3x3の変換マトリックスを表す指定された６つの数字でグラフィックコンテキストを変換します。

変換マトリックスは次のように記述されます。

ab0
cd0
ef 1

パラメーター

- **a**: **number**
  １行１列目の値。
- **b**: **number**
  １行２列目の値。
- **c**: **number**
  ２行１列目の値。
- **d**: **number**
  ２行２列目の値。
- **e**: **number**
  ３行１列目の値。
- **f**: **number**
  ３行２列目の値。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

translate

```
translate(x: number, y: number): PdfPageArea
```

指定された距離でグラフィックコンテキストを平行移動します。

パラメーター

- **x**: **number**
  X軸方向に移動する距離（ポイント単位）。

- **y**: **number**
  Y軸方向に移動する距離（ポイント単位）。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

イベント

ended

ドキュメントがレンダリングされると発生します。

引数 **PdfDocumentEndedEventArgs**

pageAdded

ドキュメントに新しいページが追加されたときに発生します。

引数 **EventArgs**

PdfDocumentEndedEventArgs クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

end イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- blob
- chunks
- empty

コンストラクタ

constructor

```
constructor(chunks: Uint8Array[]): PdfDocumentEndedEventArgs
```

PdfDocumentEndedEventArgs クラスの新しいインスタンスを初期化します。

パラメーター
- **chunks**: **Uint8Array[]**
  チャンクの配列。

戻り値 **PdfDocumentEndedEventArgs**

プロパティ

- blob

ドキュメントデータを含むBlobオブジェクトを取得します。

型 **Blob**

- chunks

ドキュメントデータを含むバッファの基本配列を取得します。

型 **Uint8Array[]**

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

PdfFont クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

フォントを表します。

コンストラクタ

- constructor

プロパティ

- family
- size
- style
- weight

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(family?: string, size?: number, style?: string, weight?: string): PdfFont
```

PdfFont クラスの新しいインスタンスを初期化します。

パラメーター

- **family**: **string** OPTIONAL
  フォントのファミリ名。
- **size**: **number** OPTIONAL
  フォントのサイズ。
- **style**: **string** OPTIONAL
  フォントのスタイル。
- **weight**: **string** OPTIONAL
  フォントの重み。

戻り値 **PdfFont**

プロパティ

- family

フォントのファミリ名を取得または設定します。

フォントファミリ名を優先度順で示すカンマ区切りリスト。各フォントファミリ名は、 **registerFont** メソッドを使用して登録された名前、PDF標準フォントファミリ名 （courier、Helvetica、symbol、times、zapfdingbats）、または スーパーファミリ名（cursive、fantasy、monospace、serif、sans-serif） の 1 つです。

型 **string**

- size

フォントのサイズを取得または設定します。

型 **number**

- style

フォントのスタイルを取得または設定します。

次の値がサポートされています：'normal'、'italic'、'oblique'

型 **string**

- weight

フォントの重みを取得または設定します。

次の値がサポートされています：'normal'、'bold'、'100'、'200'、'300' '400'、'500'、'600'、'700'、'800'、'900'

型 **string**

メソッド

- clone

```
clone(): PdfFont
```

この**PdfFont** のコピーを作成します。

戻り値 **PdfFont**

- equals

```
equals(value: PdfFont): boolean
```

指定された**PdfFont** インスタンスが現在のインスタンスと等しいかどうかを判定します。

パラメーター

- **value**: **PdfFont**
  比較する**PdfFont**。

戻り値 **boolean**

PdfGradientBrush クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
基本クラス **PdfBrush**
派生クラス **PdfLinearGradientBrush**, **PdfRadialGradientBrush**
表示 ☐継承されたメンバー ☐イベント発生元

PdfLinearGradientBrush および**PdfRadialGradientBrush** クラスの 基本クラスになる抽象クラスを表します。

このクラスは、ユーザーコード内でインスタンス化することを意図していません。

コンストラクタ

- constructor

プロパティ

- opacity
- stops

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(stops?: PdfGradientStop[], opacity?: number): PdfGradientBrush
```

PdfGradientBrush クラスの新しいインスタンスを初期化します。

パラメーター

- **stops**: **PdfGradientStop**[] OPTIONAL
  このブラシに設定する**PdfGradientStop** 配列。
- **opacity**: **number** OPTIONAL
  このブラシの不透明度。

戻り値 **PdfGradientBrush**

プロパティ

- opacity

ブラシの不透明度を取得または設定します。 この値は、[0, 1]の範囲にする必要があります。 0 はブラシが 完全に透明であり、 1 はブラシが完全に不透明であることを示します。 デフォルト値は1です。

型 **number**

- stops

ブラシのグラデーション軸内の色、オフセット、および不透明度を表す**PdfGradientStop** オブジェクト の配列を取得または設定します。 デフォルト値は空の配列です。

型 **PdfGradientStop**[]

メソッド

clone

```
clone(): PdfBrush
```

この**PdfBrush** のコピーを作成します。

継承元 **PdfBrush**
戻り値 **PdfBrush**

equals

```
equals(value: PdfGradientBrush): boolean
```

指定された**PdfGradientBrush** インスタンスが現在のインスタンスと 等しいかどうかを判定します。

パラメーター

- **value**: **PdfGradientBrush**
  比較する**PdfGradientBrush**。

戻り値 **boolean**

PdfGradientStop クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

グラデーションの遷移ポイントを決定するオブジェクトを表します。

コンストラクタ

- constructor

プロパティ

- color
- offset
- opacity

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(offset?: number, color?: any, opacity?: number): PdfGradientStop
```

PdfGradientStop クラスの新しいインスタンスを初期化します。

パラメーター

- **offset**: **number** OPTIONAL
  グラデーション軸内のグラデーション途中色の場所。
- **color**: **any** OPTIONAL
  グラデーション途中色の色。 **Color** オブジェクト、または**fromString** メソッドが受け取ることができる任意の文字列。
- **opacity**: **number** OPTIONAL
  グラデーション途中色の不透明度。

戻り値 **PdfGradientStop**

プロパティ

- color

グラデーション途中色を取得または設定します。 デフォルトの色は黒です。

型 **Color**

- offset

ブラシのグラデーション軸内のグラデーション途中色の場所を取得または設定します。 この値は、[0, 1]の範囲にする必要があります。 0 はグラデーション途中色が グラデーション軸の先頭に置かれ、 1 はグラデーション途中色が グラデーション軸の最後に置かれることを示します。 デフォルト値は0です。

型 **number**

opacity

グラデーション途中色の不透明度を取得または設定します。 この値は、[0, 1]の範囲にする必要があります。 0 はグラデーション途中色が 完全に透明であり、 1 はグラデーション途中色が完全に不透明であることを示します。デフォルト値は1です。

型 **number**

メソッド

clone

```
clone(): PdfGradientStop
```

この**PdfGradientStop** のコピーを作成します。

戻り値 **PdfGradientStop**

equals

```
equals(value: PdfGradientStop): boolean
```

指定された**PdfGradientStop** インスタンスが現在のインスタンスと等しいかどうかを決定します。

パラメーター

- **value**: **PdfGradientStop**
  比較する**PdfGradientStop**。

戻り値 **boolean**

PdfLinearGradientBrush クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
基本クラス **PdfGradientBrush**
表示 ☐継承されたメンバー ☐イベント発生元

直線グラデーションで領域を塗りつぶすために使用されるブラシを表します。

コンストラクタ

- constructor

プロパティ

- opacity
- stops
- x1
- x2
- y1
- y2

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(x1: number, y1: number, x2: number, y2: number, stops: PdfGradientStop[], opacity?: number): PdfLinearGradientBrush
```

PdfLinearGradientBrush クラスの新しいインスタンスを初期化します。

パラメーター

- **x1**: **number**
  直線グラデーションの開始点のX座標。
- **y1**: **number**
  直線グラデーションの開始点のY座標。
- **x2**: **number**
  直線グラデーションの終了点のX座標。
- **y2**: **number**
  直線グラデーションの終了点のY座標。
- **stops**: **PdfGradientStop[]**
  このブラシに設定する**PdfGradientStop** 配列。
- **opacity**: **number** OPTIONAL
  このブラシの不透明度。

戻り値 **PdfLinearGradientBrush**

プロパティ

opacity

ブラシの不透明度を取得または設定します。 この値は、[0, 1]の範囲にする必要があります。 0 はブラシが 完全に透明であり、 1 はブラシが完全に不透明であることを示します。 デフォルト値は1です。

継承元 **PdfGradientBrush**
型 **number**

stops

ブラシのグラデーション軸内の色、オフセット、および不透明度を表す**PdfGradientStop** オブジェクト の配列を取得または設定します。 デフォルト値は空の配列です。

継承元 **PdfGradientBrush**
型 **PdfGradientStop**[]

x1

ページ領域座標内の直線グラデーションの開始点のX座標（ポイント単位）を取得または設定します。

型 **number**

x2

ページ領域座標内の直線グラデーションの終了点のX座標 （ポイント単位）を取得または設定します。

型 **number**

y1

ページ領域座標内の直線グラデーションの開始点のY座標（ポイント単位）を取得または設定します。

型 **number**

y2

ページ領域座標内の直線グラデーションの終了点のY座標（ポイント単位）を取得または設定します。

型 **number**

メソッド

clone

```
clone(): PdfLinearGradientBrush
```

この**PdfLinearGradientBrush** のコピーを作成します。

戻り値 **PdfLinearGradientBrush**

equals

```
equals(value: PdfLinearGradientBrush): boolean
```

指定された**PdfLinearGradientBrush** インスタンスが現在のインスタンスと等しいかどうかを決定します。

パラメーター

- **value**: **PdfLinearGradientBrush**
  比較する**PdfLinearGradientBrush**。

戻り値 **boolean**

PdfPageArea クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
派生クラス **PdfDocument**, **PdfRunningTitle**

左上隅を(0, 0)とする独自の座標系でページの領域を表します。 テキスト、画像、パス、および変換を描画するためのメソッドを提供します。

このクラスは、ユーザーコード内でインスタンス化することを意図していません。

コンストラクタ

- constructor

プロパティ

- document
- height
- lineGap
- paths
- width
- x
- y

メソッド

- drawImage
- drawSvg
- drawText
- lineHeight
- measureText
- moveDown
- moveUp
- openImage
- rotate
- scale
- transform
- translate

コンストラクタ

constructor

```
constructor(): PdfPageArea
```

PdfRunningTitle クラスの新しいインスタンスを初期化します。

戻り値 **PdfPageArea**

プロパティ

- document

ドキュメントオブジェクトを取得します。

型 **`PdfDocument`**

- height

領域の高さ（ポイント単位）を取得します。

型 **number**

- lineGap

テキスト行の間隔（ポイント単位）を取得または設定します。

デフォルト値は0です。

型 **number**

- paths

パスを描画する機能を提供するオブジェクトを取得します。

型 **`PdfPaths`**

- width

領域の幅（ポイント単位）を取得します。

型 **number**

- x

テキストまたは画像の描画に使用されるテキストフロー内の現在の点のX座標 （ポイント単位）を取得または設定します。

型 **number**

y

テキストまたは画像の描画に使用されるテキストフロー内の現在の点のY座標 （ポイント単位）を取得または設定します。

型 **number**

メソッド

drawImage

```
drawImage(src: any, x?: number, y?: number, options?: IPdfImageDrawSettings): PdfPageArea
```

指定されたオプションを使用して、JPGまたはPNG形式で画像を描画します。

xおよびyが定義されていない場合は、代わりに@see:x および@see:y が使用されます。

最後に、画像がテキストフロー内で描画された場合、このメソッドは@see:y を更新します。 したがって、後続のテキストまたは画像は、この点の下から開始されます。

パラメーター

- **src**: **any**
  画像の取得元のURLを含む文字列、Base64エンコード画像を含むデータURI、または**IPdfImage** オブジェクト。
- **x**: **number** OPTIONAL
  画像を描画するポイントのx座標（ポイント単位）。
- **y**: **number** OPTIONAL
  画像を描画するポイントのy座標（ポイント単位）。
- **options**: **IPdfImageDrawSettings** OPTIONAL
  画像描画オプションを決定します。

戻り値 **PdfPageArea**

drawSvg

```
drawSvg(url: string, x?: number, y?: number, options?: IPdfSvgDrawSettings): PdfPageArea
```

指定されたオプションを使用してSVG画像を描画します。

xおよびyが定義されていない場合は、代わりに@see:x および@see:y が使用されます。

このメソッドは、最も外側のSVG要素のwidth属性とheight属性の値を使用し、 options.widthプロパティとoptions.heightプロパティに基づいて拡大縮小率を決定します。これらの属性のいずれかを省略した場合、 拡大縮小は行われず、画像は元のサイズでレンダリングされます。

最後に、画像がテキストフロー内で描画された場合、このメソッドは@see:y を更新します。 したがって、後続のテキストまたは画像は、この点の下から開始されます。 インクリメント値は、options.heightプロパティまたは最も外側のSVG要素のheight属性によって定義されます。 どちらも指定されていない場合、@see:y は変更されません。

このメソッドは、SVG機能の一部のみをサポートしており、主にWijmo 5のチャートコントロールをレンダリングするために提供されています。

パラメーター

- **url**: **string**
  SVG画像の取得元のURLを含む文字列、またはBase64エンコードSVG画像を含むデータURI。
- **x**: **number** OPTIONAL
  画像を描画するポイントのx座標（ポイント単位）。
- **y**: **number** OPTIONAL
  画像を描画するポイントのy座標（ポイント単位）。
- **options**: **IPdfSvgDrawSettings** OPTIONAL
  SVG画像描画オプションを決定します。

戻り値 **PdfPageArea**

drawText

```
drawText(text: string, x?: number, y?: number, options?: IPdfTextDrawSettings): IPdfTextMeasurementInfo
```

指定されたオプションを使用して文字列を描画し、測定情報を返します。

options.pen、**options.brush**、または**options.font**を省略すると、 それぞれ現在のドキュメントのペン、ブラシ、またはフォントが使用されます（**`setPen`**、 **`setBrush`**、および**`setFont`** を参照）。

文字列は、左上隅がxおよびy、幅および高さが **options.width**値および**options.height**値で定義される四角形領域内に描画されます。xとyを指定しない場合は、代わりに**x** プロパティと**y** プロパティが使用されます。

領域内のテキストは、自動的に折り返されてクリッピングされます。 **options.height**が定義されておらず、テキストが本文の下端を超えた場合、 テキストに合わせて新しいページが追加されます。

最後に、**x** プロパティと**y** プロパティの値を更新します。したがって、後続のテキストまたは画像は、この点の下から開始されます（**options.continued**の値によって異なる）。

測定結果には、テキストが複数のページまたは列に分割される可能性があることが反映されません。 テキストは単一ブロックとして処理されます。

パラメーター

- **text**: **string**
  描画するテキスト。
- **x**: **number** OPTIONAL
  テキストを描画するポイントのX座標（ポイント単位）。
- **y**: **number** OPTIONAL
  テキストを描画するポイントのY座標（ポイント単位）。
- **options**: **IPdfTextDrawSettings** OPTIONAL
  テキスト描画オプションを決定します。

戻り値 **`IPdfTextMeasurementInfo`**

lineHeight

```
lineHeight(font?: PdfFont): number
```

指定されたフォントの行の高さを取得します。

フォントが指定されていない場合は、現在のドキュメントで使用されているフォントが使用されます。

パラメーター

- **font**: **PdfFont** OPTIONAL
  行の高さを取得するフォント。

戻り値 **number**

measureText

```
measureText(text: string, font?: PdfFont, options?: IPdfTextMeasurementSettings): IPdfTextMeasurementInfo
```

指定されたフォントおよびテキスト描画オプションを使用して、テキストをレンダリングせずに測定します。

フォントが指定されていない場合は、現在のドキュメントで使用されているフォントが使用されます。

このメソッドは、**drawText** と同じテキストレンダリングエンジンを使用するため、 options.widthが指定されていない場合は、同様に@see:x およびページの右マージンに制約されます。測定結果には、テキストが複数のページまたは列に分割される可能性があることが反映されません。 テキストは単一ブロックとして処理されます。

パラメーター

- **text**: **string**
  測定するテキスト。
- **font**: **PdfFont** OPTIONAL
  テキストに適用されるフォント。
- **options**: **IPdfTextMeasurementSettings** OPTIONAL
  テキスト描画オプションを決定します。

戻り値 **IPdfTextMeasurementInfo**

moveDown

```
moveDown(lines?: number, font?: PdfFont): PdfPageArea
```

指定されたフォントまたはフォントが指定されていない場合は現在のドキュメントのフォントを使用して、指定された行数だけ@see:y を下に移動します。

パラメーター

- **lines**: **number** OPTIONAL
  下に移動する行数。
- **font**: **PdfFont** OPTIONAL
  行の高さを計算するフォント。

戻り値 **PdfPageArea**

moveUp

```
moveUp(lines?: number, font?: PdfFont): PdfPageArea
```

指定されたフォントまたはフォントが指定されていない場合は現在のドキュメントのフォントを使用して、指定された行数だけ@see:y を上に移動します。

パラメーター

- **lines**: **number** OPTIONAL
  上に移動する行数。
- **font**: **PdfFont** OPTIONAL
  行の高さを計算するフォント。

戻り値 **PdfPageArea**

openImage

```
openImage(url: string): IPdfImage
```

JPGまたはPNG形式で画像を開きます。

パラメーター

- **url**: **string**
  画像の取得元のURLを含む文字列、またはBase64エンコード画像を含むデータURI。

戻り値 **IPdfImage**

rotate

```
rotate(angle: number, origin?: Point): PdfPageArea
```

指定された角度でグラフィックコンテキストを時計回りに回転します。

パラメーター

- **angle**: **number**
  回転角度（度単位）。

- **origin**: **Point** OPTIONAL
  回転の中心の**Point** （ポイント単位）。 指定しない場合は、左上隅が使用されます。

戻り値 **PdfPageArea**

scale

```
scale(xFactor: number, yFactor?: number, origin?: Point): PdfPageArea
```

指定された拡大率でグラフィックコンテキストを拡大縮小します。

範囲[0, 1]内の拡大率の値は、サイズが小さくなることを示します。 1 より大きい拡大率の値は、サイズが大きくなることを示します。

パラメーター

- **xFactor**: **number**
  Xサイズを拡大縮小する係数。

- **yFactor**: **number** OPTIONAL
  Yサイズを拡大縮小する係数。 指定しない場合は、xFactorと等しい値と見なされます。

- **origin**: **Point** OPTIONAL
  拡大縮小の中心となる**Point** （ポイント単位）。 指定しない場合は、左上隅が使用されます。

戻り値 **PdfPageArea**

transform

```
transform(a: number, b: number, c: number, d: number, e: number, f: number): PdfPageArea
```

3x3の変換マトリックスを表す指定された６つの数字でグラフィックコンテキストを変換します。

変換マトリックスは次のように記述されます。

ab0
cd0
ef 1

パラメーター

- **a**: **number**
  １行１列目の値。
- **b**: **number**
  １行２列目の値。
- **c**: **number**
  ２行１列目の値。
- **d**: **number**
  ２行２列目の値。
- **e**: **number**
  ３行１列目の値。
- **f**: **number**
  ３行２列目の値。

戻り値 **PdfPageArea**

translate

```
translate(x: number, y: number): PdfPageArea
```

指定された距離でグラフィックコンテキストを平行移動します。

パラメーター

- **x**: **number**
  X軸方向に移動する距離（ポイント単位）。
- **y**: **number**
  Y軸方向に移動する距離（ポイント単位）。

戻り値 **PdfPageArea**

PdfPaths クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

グラフィックパスを作成し、それらを描画したりクリッピング用に使用するためのメソッドを提供します。

パス作成メソッドの呼び出しは、**stroke** 、**fill**、 **fillAndStroke**、または**clip** メソッドで終了する必要があります。 パス作成/描画/クリッピングに直接適用されないドキュメントメソッド （ペンの変更、テキストの描画、グラフィック状態の保存など）は、パスが終了されるまで 使用が禁止されます。
lineTo、**bezierCurveTo**、および**quadraticCurveTo** メソッドで パスを開始することはできません。これらのメソッドの前に**moveTo** が必要です。

これらのメソッドは連結できます。

```
doc.paths.moveTo(0, 0).lineTo(100, 100).stroke();
```

このクラスは、ユーザーコード内でインスタンス化することを意図していません。

コンストラクタ

- constructor

メソッド

- bezierCurveTo
- circle
- clip
- closePath
- ellipse
- fill
- fillAndStroke
- lineTo
- moveTo
- polygon
- quadraticCurveTo
- rect
- roundedRect
- stroke
- svgPath

コンストラクタ

constructor

```
constructor(doc: PdfDocument, offset: Point): PdfPaths
```

PdfPaths クラスの新しいインスタンスを初期化します。

パラメーター

- **doc**: **PdfDocument**
  ドキュメント。
- **offset**: **Point**
  オフセット

戻り値 **PdfPaths**

メソッド

bezierCurveTo

```
bezierCurveTo(cp1x: number, cp1y: number, cp2x: number, cp2y: number, x: number, y: number): PdfPaths
```

現在の点から新しい点までベジェ曲線を描画します。制御点として(cp1x, cp1y) および(cp2x, cp2y)を使用します。

(x, y)が新しい現在の点になります。

パラメーター

- **cp1x**: **number**
  最初の制御点のX座標（ポイント単位）。
- **cp1y**: **number**
  最初の制御点のY座標（ポイント単位）。
- **cp2x**: **number**
  ２番目の制御点のX座標（ポイント単位）。
- **cp2y**: **number**
  ２番目の制御点のY座標（ポイント単位）。
- **x**: **number**
  新しい点のX座標（ポイント単位）。
- **y**: **number**
  新しい点のY座標（ポイント単位）。

戻り値 **PdfPaths**

circle

```
circle(x: number, y: number, radius: number): PdfPaths
```

円を描画します。

パラメーター

- **x**: **number**
  円の中心のX座標（ポイント単位）。
- **y**: **number**
  円の中心のY座標（ポイント単位）。
- **radius**: **number**
  円の半径（ポイント単位）。

戻り値 **PdfPaths**

clip

```
clip(rule?: PdfFillRule): PdfPaths
```

描画操作が作用するページの領域を制限するために使用されるクリッピングパスを作成します。

パラメーター

- **rule**: **PdfFillRule** OPTIONAL
  使用する塗りつぶしルール。

戻り値 **PdfPaths**

closePath

```
closePath(): PdfPaths
```

現在のパスの現在の点から最初の点まで線を描画して、現在のパスを閉じます。

戻り値 **PdfPaths**

ellipse

```
ellipse(x: number, y: number, radiusX: number, radiusY?: number): PdfPaths
```

楕円を描画します。

パラメーター

- **x**: **number**
  楕円の中心のX座標（ポイント単位）。
- **y**: **number**
  楕円の中心のY座標（ポイント単位）。
- **radiusX**: **number**
  X軸方向の楕円の半径（ポイント単位）。
- **radiusY**: **number** OPTIONAL
  Y軸方向の楕円の半径（ポイント単位）。 指定しない場合は、radiusXと同じと見なされます。

戻り値 **PdfPaths**

fill

```
fill(brushOrColor?: any, rule?: PdfFillRule): PdfPaths
```

指定されたブラシおよびルールでパスの塗りつぶしを行います。 ブラシを指定しない場合は、デフォルトのドキュメントブラシが使用されます（**setBrush** メソッドを参照）。

brushOrColor引数は、次の値を取ることができます。

- **PdfBrush** オブジェクト。
- **Color** オブジェクト、または**fromString** メソッドが受け取ることができる任意の文字列。 この場合、指定された色の**PdfBrush** オブジェクトが内部的に作成されます。

パラメーター

- **brushOrColor**: **any** OPTIONAL
  使用するブラシまたは色。

- **rule**: **PdfFillRule** OPTIONAL
  使用する塗りつぶしルール。

戻り値 **PdfPaths**

fillAndStroke

```
fillAndStroke(brushOrColor?: any, penOrColor?: any, rule?: PdfFillRule): PdfPaths
```

指定されたブラシ、ペン、およびルールでパスの塗りつぶしとストロークを行います。 ブラシおよびペンを指定しない場合は、デフォルトのドキュメントブラシおよびペンが使用されます （**setBrush** 、**setPen** メソッドを参照）。

brushOrColor引数は、次の値を取ることができます。

- **PdfBrush** オブジェクト。
- **Color** オブジェクト、または**fromString** メソッドが受け取ることができる任意の文字列。 この場合、指定された色の**PdfBrush** オブジェクトが内部的に作成されます。

penOrColor引数は、次の値を取ることができます。

- **PdfPen** オブジェクト。
- **Color** オブジェクト、または**fromString** メソッドが受け取ることができる任意の文字列。 この場合、指定された色の**PdfPen** オブジェクトが内部的に作成されます。

パラメーター

- **brushOrColor**: **any** OPTIONAL
  使用するブラシまたは色。

- **penOrColor**: **any** OPTIONAL
  使用するペンまたは色。

- **rule**: **PdfFillRule** OPTIONAL
  使用する塗りつぶしルール。

戻り値 **PdfPaths**

lineTo

```
lineTo(x: number, y: number): PdfPaths
```

現在の点から新しい点まで線を描画します。

(x, y)が新しい現在の点になります。

パラメーター

- **x**: **number**
  新しい点のX座標（ポイント単位）。
- **y**: **number**
  新しい点のY座標（ポイント単位）。

戻り値 **PdfPaths**

moveTo

```
moveTo(x: number, y: number): PdfPaths
```

新しい現在の点を設定します。

パラメーター

- **x**: **number**
  新しい点のX座標（ポイント単位）。
- **y**: **number**
  新しい点のY座標（ポイント単位）。

戻り値 **PdfPaths**

polygon

```
polygon(points: number[][]): PdfPaths
```

指定された点の配列を使用して多角形を描画します。

パラメーター

- **points**: **number[][]**
  点のXおよびY座標（ポイント単位）を指定する２要素配列[x, y]。

戻り値 **PdfPaths**

quadraticCurveTo

```
quadraticCurveTo(cpx: number, cpy: number, x: number, y: number): PdfPaths
```

現在の点から新しい点まで２次曲線を描画します。制御点として現在の点 および(cpx, cpy)を使用します。

(x, y)が新しい現在の点になります。

パラメーター

- **cpx**: **number**
  制御点のX座標（ポイント単位）。
- **cpy**: **number**
  制御点のY座標（ポイント単位）。
- **x**: **number**
  新しい点のX座標（ポイント単位）。
- **y**: **number**
  新しい点のY座標（ポイント単位）。

戻り値 **PdfPaths**

rect

```
rect(x: number, y: number, width: number, height: number): PdfPaths
```

四角形を描画します。

パラメーター

- **x**: **number**
  四角形の左上隅のX座標（ポイント単位）。
- **y**: **number**
  四角形の左上隅のY座標（ポイント単位）。
- **width**: **number**
  四角形の幅（ポイント単位）。
- **height**: **number**
  四角形の幅（ポイント単位）。

戻り値 **PdfPaths**

roundedRect

```
roundedRect(x: number, y: number, width: number, height: number, cornerRadius?: number): PdfPaths
```

角丸四角形を描画します。

パラメーター

- **x**: **number**
  四角形の左上隅のX座標（ポイント単位）。
- **y**: **number**
  四角形の左上隅のY座標（ポイント単位）。
- **width**: **number**
  四角形の幅（ポイント単位）。
- **height**: **number**
  四角形の幅（ポイント単位）。
- **cornerRadius**: **number** OPTIONAL
  四角形の角の丸み（ポイント単位）。 デフォルト値は0です。

戻り値 **PdfPaths**

stroke

```
stroke(penOrColor?: any): PdfPaths
```

指定されたペンでパスをストロークします。 ペンを指定しない場合は、デフォルトのドキュメントペンが使用されます （**setPen** メソッドを参照）。

penOrColor引数は、次の値を取ることができます。

- **PdfPen** オブジェクト。
- **Color** オブジェクト、または**fromString** メソッドが受け取ることができる任意の文字列。 この場合、指定された色の**PdfPen** オブジェクトが内部的に作成されます。

パラメーター

- **penOrColor**: **any** OPTIONAL
  使用するペンまたは色。

戻り値 **PdfPaths**

svgPath

```
svgPath(path: string): PdfPaths
```

SVG 1.1パスを描画します。

パラメーター

- **path**: **string**
  描画するSVGパス。

戻り値 **PdfPaths**

PdfPen クラス

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

パスおよびテキストをストロークするために使用されるオブジェクトを決定します。

コンストラクタ

- constructor

プロパティ

- brush
- cap
- color
- dashPattern
- join
- miterLimit
- width

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(colorOrBrushOrOptions?: any, width?: number, dashPattern?: PdfDashPattern, cap?: PdfLineCapStyle, join?: PdfLineJoinStyle,
miterLimit?: number): PdfPen
```

指定された色、ブラシ、またはJavaScriptオブジェクトを使用して、**PdfPen** クラスの新しいインスタンスを初期化します。

最初の引数は、次の値を取ることができます。

- **Color** オブジェクト、または**fromString** メソッドが受け取ることができる任意の文字列。
- **PdfBrush** オブジェクト。
- 初期化プロパティを含むJavaScriptオブジェクト（他のすべての引数は無視されます）。

パラメーター

- **colorOrBrushOrOptions**: **any** OPTIONAL
  使用する色、ブラシ、またはJavaScriptオブジェクト。
- **width**: **number** OPTIONAL
  使用する幅。
- **dashPattern**: **PdfDashPattern** OPTIONAL
  使用する破線パターン。
- **cap**: **PdfLineCapStyle** OPTIONAL
  使用するラインキャップスタイル。
- **join**: **PdfLineJoinStyle** OPTIONAL
  使用する線の結合スタイル。
- **miterLimit**: **number** OPTIONAL
  使用するマイターリミット。

戻り値 **PdfPen**

プロパティ

brush

パスをストロークするために使用されるブラシを取得または設定します。 **color** プロパティが定義されている場合は、それより優先されます。

型 **PdfBrush**

cap

ストロークパスの開いた終端に使用される形状を取得または設定します。 デフォルト値は**Butt**です。

型 **PdfLineCapStyle**

color

パスをストロークするために使用される色を取得または設定します。 デフォルトの色は黒です。

型 **Color**

dashPattern

パスのストロークに使用される破線パターンを取得します。 デフォルト値は実線です。

型 **PdfDashPattern**

- join

ストロークパスの角に使用される形状を取得または設定します。 デフォルト値は**Miter**です。

型 **PdfLineJoinStyle**

- miterLimit

マイター長がどの程度長くなったら線の結合がマイターからベベルに変わるかを 線の幅に対する比率の最大値で指定します。 デフォルト値は10です。

型 **number**

- width

パスをストロークするために使用される線の幅（ポイント単位）を取得または設定します。 デフォルトの幅は 1 です。

型 **number**

メソッド

- clone

```
clone(): PdfPen
```

この**PdfPen** のコピーを作成します。

戻り値 **PdfPen**

- equals

```
equals(value: PdfPen): boolean
```

指定された**PdfPen** インスタンスが現在のインスタンスと等しいかどうかを判定します。

パラメーター

- **value**: **PdfPen**
  比較する**PdfPen**。

戻り値 **boolean**

PdfRadialGradientBrush クラス

ファイル	`wijmo.pdf.js`
モジュール	**wijmo.pdf**
基本クラス	**PdfGradientBrush**
表示	☐ 継承されたメンバー ☐ イベント発生元

放射状グラデーションで領域を塗りつぶすために使用されるブラシを表します。

コンストラクタ

- constructor

プロパティ

- opacity
- r1
- r2
- stops
- x1
- x2
- y1
- y2

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(x1: number, y1: number, r1: number, x2: number, y2: number, r2: number, stops: PdfGradientStop[], opacity?: number): PdfRadialGradientBrush
```

PdfRadialGradientBrush クラスの新しいインスタンスを初期化します。

パラメーター

- **x1**: **number**
  放射状グラデーションの内円の中心のX座標。
- **y1**: **number**
  放射状グラデーションの内円の中心のY座標。
- **r1**: **number**
  放射状グラデーションの内円の半径。
- **x2**: **number**
  放射状グラデーションの外円の中心のX座標
- **y2**: **number**
  放射状グラデーションの外円の中心のY座標。
- **r2**: **number**
  放射状グラデーションの外円の半径。
- **stops**: **PdfGradientStop[]**
  このブラシに設定する**PdfGradientStop** 配列。
- **opacity**: **number** OPTIONAL
  このブラシの不透明度。

戻り値 **PdfRadialGradientBrush**

プロパティ

opacity

ブラシの不透明度を取得または設定します。 この値は、[0, 1]の範囲にする必要があります。 0 はブラシが 完全に透明であり、 1 はブラシが完全に不透明であることを示します。 デフォルト値は1です。

継承元 **PdfGradientBrush**
型 **number**

r1

ページ領域座標内の放射状グラデーションの開始点を表す内円の半径 （ポイント単位）を取得または設定します。

型 **number**

r2

ページ領域座標内の放射状グラデーションの終了点を表す外円の半径（ポイント単位）を取得または設定します。

型 **number**

- stops

ブラシのグラデーション軸内の色、オフセット、および不透明度を表す**PdfGradientStop** オブジェクト の配列を取得または設定します。 デフォルト値は空の配列です。

継承元 **PdfGradientBrush**
型 **PdfGradientStop**[]

- x1

ページ領域座標内の放射状グラデーションの開始点を表す内円の中心のX座標 （ポイント単位）を取得または設定します。

型 **number**

- x2

ページ領域座標内の放射状グラデーションの終了点を表す外円の中心のX座標（ポイント単位）を取得または設定します。

型 **number**

- y1

ページ領域座標内の放射状グラデーションの開始点を表す内円の中心のY座標 （ポイント単位）を取得または設定します。

型 **number**

- y2

ページ領域座標内の放射状グラデーションの終了点を表す外円の中心のY座標（ポイント単位）を取得または設定します。

型 **number**

メソッド

- clone

```
clone(): PdfRadialGradientBrush
```

この**PdfRadialGradientBrush** のコピーを作成します。

戻り値 **PdfRadialGradientBrush**

- equals

```
equals(value: PdfRadialGradientBrush): boolean
```

指定された**PdfRadialGradientBrush** インスタンスが現在のインスタンスと等しいかどうかを判定します。

パラメーター
- **value**: **PdfRadialGradientBrush**
  比較する**PdfRadialGradientBrush**。

戻り値 **boolean**

PdfRunningTitle クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
基本クラス **PdfPageArea**
表示 ☐継承されたメンバー ☐イベント発生元

ヘッダー、フッターなどページの欄外見出しを表します。

このクラスは、ユーザーコード内でインスタンス化することを意図していません。

コンストラクタ

- constructor

プロパティ

- declarative
- document
- height
- lineGap
- paths
- width
- x
- y

メソッド

- drawImage
- drawSvg
- drawText
- lineHeight
- measureText
- moveDown
- moveUp
- openImage
- rotate
- scale
- transform
- translate

コンストラクタ

constructor

```
constructor(options?: any): PdfRunningTitle
```

PdfRunningTitle クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  初期化設定を含むオプションのオブジェクト。

戻り値 **PdfRunningTitle**

プロパティ

declarative

欄外見出しのコンテンツを宣言的に設定するための機能を提供するオブジェクトを取得または設定します。

型 **PdfRunningTitleDeclarativeContent**

document

ドキュメントオブジェクトを取得します。

継承元 **PdfPageArea**
型 **PdfDocument**

height

欄外見出しの高さ（ポイント単位）を取得または設定します。 欄外見出しを非表示にするには、このプロパティを 0 に設定します。 このプロパティを変更しても、前の描画に影響はなく、サイズ変更も クリップもされません。

デフォルト値は24です。

型 **number**

lineGap

テキスト行の間隔（ポイント単位）を取得または設定します。

デフォルト値は0です。

継承元 **PdfPageArea**
型 **number**

paths

パスを描画する機能を提供するオブジェクトを取得します。

継承元 **PdfPageArea**
型 **PdfPaths**

width

領域の幅（ポイント単位）を取得します。

継承元 **PdfPageArea**
型 **number**

x

テキストまたは画像の描画に使用されるテキストフロー内の現在の点のX座標 （ポイント単位）を取得または設定します。

継承元 **PdfPageArea**
型 **number**

y

テキストまたは画像の描画に使用されるテキストフロー内の現在の点のY座標 （ポイント単位）を取得または設定します。

継承元 **PdfPageArea**
型 **number**

メソッド

drawImage

```
drawImage(src: any, x?: number, y?: number, options?: IPdfImageDrawSettings): PdfPageArea
```

指定されたオプションを使用して、JPGまたはPNG形式で画像を描画します。

xおよびyが定義されていない場合は、代わりに@see:x および@see:y が使用されます。

最後に、画像がテキストフロー内で描画された場合、このメソッドは@see:y を更新します。 したがって、後続のテキストまたは画像は、この点の下から開始されます。

パラメーター

- **src**: **any**
  画像の取得元のURLを含む文字列、Base64エンコード画像を含むデータURI、または**IPdfImage** オブジェクト。
- **x**: **number** OPTIONAL
  画像を描画するポイントのx座標（ポイント単位）。
- **y**: **number** OPTIONAL
  画像を描画するポイントのy座標（ポイント単位）。
- **options**: **IPdfImageDrawSettings** OPTIONAL
  画像描画オプションを決定します。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

drawSvg

```
drawSvg(url: string, x?: number, y?: number, options?: IPdfSvgDrawSettings): PdfPageArea
```

指定されたオプションを使用してSVG画像を描画します。

xおよびyが定義されていない場合は、代わりに@see:x および@see:y が使用されます。

このメソッドは、最も外側のSVG要素のwidth属性とheight属性の値を使用し、 options.widthプロパティとoptions.heightプロパティに基づいて拡大縮小率を決定します。これらの属性のいずれかを省略した場合、 拡大縮小は行われず、画像は元のサイズでレンダリングされます。

最後に、画像がテキストフロー内で描画された場合、このメソッドは@see:y を更新します。 したがって、後続のテキストまたは画像は、この点の下から開始されます。 インクリメント値は、options.heightプロパティまたは最も外側のSVG要素のheight属性によって定義されます。 どちらも指定されていない場合、@see:y は変更されません。

このメソッドは、SVG機能の一部のみをサポートしており、主にWijmo 5のチャートコントロールをレンダリングするために提供されています。

パラメーター

- **url**: **string**
  SVG画像の取得元のURLを含む文字列、またはBase64エンコードSVG画像を含むデータURI。
- **x**: **number** OPTIONAL
  画像を描画するポイントのx座標（ポイント単位）。
- **y**: **number** OPTIONAL
  画像を描画するポイントのy座標（ポイント単位）。
- **options**: **IPdfSvgDrawSettings** OPTIONAL
  SVG画像描画オプションを決定します。

継承元 `PdfPageArea`
戻り値 `PdfPageArea`

drawText

```
drawText(text: string, x?: number, y?: number, options?: IPdfTextDrawSettings): IPdfTextMeasurementInfo
```

指定されたオプションを使用して文字列を描画し、測定情報を返します。

options.pen、**options.brush**、または**options.font**を省略すると、 それぞれ現在のドキュメントのペン、ブラシ、またはフォントが使用されます（**setPen**、 **setBrush**、および**setFont** を参照）。

文字列は、左上隅がxおよびy、幅および高さが **options.width**値および**options.height**値で定義される四角形領域内に描画されます。xとyを指定しない場合は、代わりに**x** プロパティと**y** プロパティが使用されます。

領域内のテキストは、自動的に折り返されてクリッピングされます。 **options.height**が定義されておらず、テキストが本文の下端を超えた場合、 テキストに合わせて新しいページが追加されます。

最後に、**x** プロパティと**y** プロパティの値を更新します。したがって、後続のテキストまたは画像は、この点の下から開始されます（**options.continued**の値によって異なる）。

測定結果には、テキストが複数のページまたは列に分割される可能性があることが反映されません。 テキストは単一ブロックとして処理されます。

パラメーター

- **text**: **string**
  描画するテキスト。
- **x**: **number** OPTIONAL
  テキストを描画するポイントのX座標（ポイント単位）。
- **y**: **number** OPTIONAL
  テキストを描画するポイントのY座標（ポイント単位）。
- **options**: **IPdfTextDrawSettings** OPTIONAL
  テキスト描画オプションを決定します。

継承元 **PdfPageArea**
戻り値 **IPdfTextMeasurementInfo**

lineHeight

```
lineHeight(font?: PdfFont): number
```

指定されたフォントの行の高さを取得します。

フォントが指定されていない場合は、現在のドキュメントで使用されているフォントが使用されます。

パラメーター

- **font**: **PdfFont** OPTIONAL
  行の高さを取得するフォント。

継承元 **PdfPageArea**
戻り値 **number**

measureText

```
measureText(text: string, font?: PdfFont, options?: IPdfTextMeasurementSettings): IPdfTextMeasurementInfo
```

指定されたフォントおよびテキスト描画オプションを使用して、テキストをレンダリングせずに測定します。

フォントが指定されていない場合は、現在のドキュメントで使用されているフォントが使用されます。

このメソッドは、**drawText** と同じテキストレンダリングエンジンを使用するため、 options.widthが指定されていない場合は、同様に@see:x およびページの右マージンに制約されます。測定結果には、テキストが複数のページまたは列に分割される可能性があることが反映されません。 テキストは単一ブロックとして処理されます。

パラメーター

- **text**: **string**
  測定するテキスト。
- **font**: **PdfFont** OPTIONAL
  テキストに適用されるフォント。
- **options**: **IPdfTextMeasurementSettings** OPTIONAL
  テキスト描画オプションを決定します。

継承元 **PdfPageArea**
戻り値 **IPdfTextMeasurementInfo**

moveDown

```
moveDown(lines?: number, font?: PdfFont): PdfPageArea
```

指定されたフォントまたはフォントが指定されていない場合は現在のドキュメントのフォントを使用して、指定された行数だけ@see:y を下に移動します。

パラメーター

- **lines**: **number** OPTIONAL
  下に移動する行数。
- **font**: **PdfFont** OPTIONAL
  行の高さを計算するフォント。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

moveUp

```
moveUp(lines?: number, font?: PdfFont): PdfPageArea
```

指定されたフォントまたはフォントが指定されていない場合は現在のドキュメントのフォントを使用して、指定された行数だけ@see:y を上に移動します。

パラメーター

- **lines**: **number** OPTIONAL
  上に移動する行数。
- **font**: **PdfFont** OPTIONAL
  行の高さを計算するフォント。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

openImage

```
openImage(url: string): IPdfImage
```

JPGまたはPNG形式で画像を開きます。

パラメーター

- **url**: **string**
  画像の取得元のURLを含む文字列、またはBase64エンコード画像を含むデータURI。

継承元 **PdfPageArea**
戻り値 **IPdfImage**

rotate

```
rotate(angle: number, origin?: Point): PdfPageArea
```

指定された角度でグラフィックコンテキストを時計回りに回転します。

パラメーター

- **angle**: **number**
  回転角度（度単位）。

- **origin**: **Point** OPTIONAL
  回転の中心の**Point** （ポイント単位）。 指定しない場合は、左上隅が使用されます。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

scale

```
scale(xFactor: number, yFactor?: number, origin?: Point): PdfPageArea
```

指定された拡大率でグラフィックコンテキストを拡大縮小します。

範囲[0, 1]内の拡大率の値は、サイズが小さくなることを示します。 1 より大きい拡大率の値は、サイズが大きくなることを示します。

パラメーター

- **xFactor**: **number**
  Xサイズを拡大縮小する係数。

- **yFactor**: **number** OPTIONAL
  Yサイズを拡大縮小する係数。 指定しない場合は、xFactorと等しい値と見なされます。

- **origin**: **Point** OPTIONAL
  拡大縮小の中心となる**Point** （ポイント単位）。 指定しない場合は、左上隅が使用されます。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

transform

```
transform(a: number, b: number, c: number, d: number, e: number, f: number): PdfPageArea
```

3x3の変換マトリックスを表す指定された６つの数字でグラフィックコンテキストを変換します。

変換マトリックスは次のように記述されます。

ab0
cd0
ef 1

パラメーター

- **a**: **number**
  １行１列目の値。
- **b**: **number**
  １行２列目の値。
- **c**: **number**
  ２行１列目の値。
- **d**: **number**
  ２行２列目の値。
- **e**: **number**
  ３行１列目の値。
- **f**: **number**
  ３行２列目の値。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

translate

```
translate(x: number, y: number): PdfPageArea
```

指定された距離でグラフィックコンテキストを平行移動します。

パラメーター

- **x**: **number**
  X軸方向に移動する距離（ポイント単位）。
- **y**: **number**
  Y軸方向に移動する距離（ポイント単位）。

継承元 **PdfPageArea**
戻り値 **PdfPageArea**

PdfRunningTitleDeclarativeContent クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

欄外見出しの宣言型コンテンツを表します。

コンストラクタ

- constructor

プロパティ

- brush
- font
- text

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(text?: string, font?: PdfFont, brushOrColor?: any): PdfRunningTitleDeclarativeContent
```

PdfRunningTitleDeclarativeContent クラスの新しいインスタンスを初期化します。

パラメーター

- **text**: **string** OPTIONAL
  欄外見出しのテキスト。
- **font**: **PdfFont** OPTIONAL
  テキストのフォント。
- **brushOrColor**: **any** OPTIONAL
  テキストの塗りつぶしに使用される**PdfBrush**、**Color** 、または**fromString** メソッドが受け取ることができる任意の文字列。

戻り値 **PdfRunningTitleDeclarativeContent**

プロパティ

- brush

text の塗りつぶしに使用されるブラシを取得または設定します。

型 **PdfBrush**

- font

text のフォントを取得または設定します。

型 **PdfFont**

- text

欄外見出しのテキストを取得または設定します。

最大3つのタブ文字（'\t'）を入れてテキストをいくつかの部分に 分割し、各部分をページ領域内で左揃え、中央揃え、および右揃えに整列できます。'&[Page]'と'&[Pages]'の2つのマクロがサポートされます。前者は 現在のページインデックスを示し、後者はページ数を示します。

たとえば、10ページあるドキュメントの最初のページの場合、次の指定は

```
'&[Page]\\&[Pages]\theader\t&[Page]\\&[Pages]'
```

次のように変換されます。

```
'1\10 header 1\10'
```

型 **string**

メソッド

- clone

```
clone(): PdfRunningTitleDeclarativeContent
```

この**PdfRunningTitleDeclarativeContent** のコピーを作成します。

戻り値 **PdfRunningTitleDeclarativeContent**

- equals

```
equals(value: PdfRunningTitleDeclarativeContent): boolean
```

指定された**PdfRunningTitleDeclarativeContent** インスタンスが 現在のインスタンスと等しいかどうかを判定します。

パラメーター

- **value**: **PdfRunningTitleDeclarativeContent**
  比較する**PdfRunningTitleDeclarativeContent**。

戻り値 **boolean**

PdfSolidBrush クラス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
基本クラス **PdfBrush**
表示 ☐継承されたメンバー ☐イベント発生元

色で領域を塗りつぶすために使用されるブラシを表します。

コンストラクタ

- constructor

プロパティ

- color

メソッド

- clone
- equals

コンストラクタ

constructor

```
constructor(color?: any): PdfSolidBrush
```

PdfSolidBrush クラスの新しいインスタンスを初期化します。

パラメーター

- **color**: **any** OPTIONAL
  このブラシの色。 **Color** オブジェクト、または**fromString** メソッドが受け取ることができる任意の文字列。

戻り値 **PdfSolidBrush**

プロパティ

- color

ブラシの色を取得または設定します。 デフォルトの色は黒です。

型 **Color**

メソッド

- clone

```
clone(): PdfSolidBrush
```

この**PdfSolidBrush** のコピーを作成します。

戻り値 **PdfSolidBrush**

equals

```
equals(value: PdfSolidBrush): boolean
```

指定された**PdfSolidBrush** インスタンスが現在のインスタンスと等しいかどうかを判定します。

パラメーター

- **value**: **PdfSolidBrush**
  比較する**PdfSolidBrush**。

戻り値 **boolean**

IPdfBufferedPageRange インターフェイス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

bufferedPageRange メソッドから返されるバッファページの範囲を表します。

プロパティ

- count
- start

プロパティ

- count

バッファページの数を決定します。

型 **number**

- start

最初のバッファページの０から始まるインデックスを決定します。

型 **number**

IPdfDocumentInfo インターフェイス

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

info プロパティで使用されるドキュメント情報を表します。

プロパティ

- author
- creationDate
- keywords
- modDate
- subject
- title

プロパティ

author

ドキュメントの作成者の名前を決定します。

型 **string**

creationDate

ドキュメントの作成日時を決定します。

型 **Date**

keywords

ドキュメントに関連付けられているキーワードを決定します。

型 **string**

modDate

ドキュメントが最後に変更された日時を決定します。

型 **Date**

subject

ドキュメントのサブタイトルを決定します。

型 **string**

title

ドキュメントのタイトルを決定します。

型 **string**

IPdfFontAttributes インターフェイス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

フォント属性を表します。

プロパティ

- cursive
- fantasy
- monospace
- sansSerif
- serif

プロパティ

cursive

グリフは、仕上げストローク、フレア付きまたは先細の終端、または実際のセリフ付き終端を持ちます。

型 **boolean**

fantasy

Fantasyフォントは、文字を面白く表現した装飾的なフォントです。

型 **boolean**

monospace

すべてのグリフは同じ幅を持ちます。

型 **boolean**

sansSerif

グリフは飾りのないストローク終端を持ちます。

型 **boolean**

serif

グリフは、仕上げストローク、フレア付きまたは先細の終端、または実際のセリフ付き終端を持ちます。

型 **boolean**

IPdfFontFile インターフェイス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
インターフェイス **IPdfFontAttributes**

registerFont および**registerFontAsync** メソッドによって登録される フォントの設定を表します。

プロパティ

- family
- name
- source
- style
- weight

プロパティ

- family

TrueTypeコレクションまたはDatafork TrueType フォントファミリを決定するオプションパラメータ。

型 **string**

- name

使用するフォントの名前。

型 **string**

- source

フォントのロード元のバイナリデータまたはURLを含むArrayBuffer。 TrueType（.ttf）、TrueType Collection（.ttc）、Datafork TrueType（.dfont）の各フォント形式がサポートされています。

型 **any**

- style

フォントのスタイル。次の値のいずれか：'normal'、'italic'、'oblique'

型 **string**

- weight

フォントの重み。次の値のいずれか：'normal'、'bold'、'100'、'200'、 '300'、'400'、'500'、'600'、'700'、'800'、'900'

型 **string**

IPdfImage インターフェイス

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

openImage メソッドを使用して開いた画像を表します。

プロパティ

- height
- width

プロパティ

- height

画像の高さ（ピクセル単位）。

型 **number**

- width

画像の幅（ピクセル単位）。

型 **number**

IPdfImageDrawSettings インターフェイス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

drawImage メソッドで使用される画像描画設定を表します。

幅と高さのオプションをどちらも指定しない場合、画像は元のサイズで レンダリングされます。幅だけを指定した場合、画像は指定された幅 に合わせて比例して拡大縮小されます。高さだけを指定した場合、画像は指定された高さ に合わせて比例して拡大縮小されます。幅と高さの両方を 指定した場合、画像は、stretchProportionallyプロパティに基づいてそのサイズまで 引き伸ばされます。

プロパティ

- align
- height
- stretchProportionally
- vAlign
- width

プロパティ

- align

比例引き伸ばしの場合の水平方向の配置を決定します。

型 **PdfImageHorizontalAlign**

- height

画像の高さ（ポイント単位）を決定します。

型 **number**

- stretchProportionally

幅と高さの両方のオプションが指定された場合に、画像が比例して 引き伸ばされるかどうかを示します。

型 **boolean**

- vAlign

比例引き伸ばしの場合の垂直方向の配置を決定します。

型 **PdfImageVerticalAlign**

- width

画像の幅（ポイント単位）を決定します。

型 **number**

IPdfPageMargins インターフェイス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

ページマージンを表します。

プロパティ

- bottom
- left
- right
- top

プロパティ

- bottom

下マージン（ポイント単位）を決定します。

型 **number**

- left

左マージン（ポイント単位）を決定します。

型 **number**

- right

右マージン（ポイント単位）を決定します。

型 **number**

- top

上マージン（ポイント単位）を決定します。

型 **number**

IPdfPageSettings インターフェイス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

ページ設定を表します。

プロパティ

- layout
- margins
- size

プロパティ

- layout

ページレイアウトを決定します。

型 **PdfPageOrientation**

- margins

ページマージンを決定します。

型 **IPdfPageMargins**

- size

ページサイズを決定します。 以下の値がサポートされます。

- **PdfPageSize** :定義済みサイズ。
- **Size** :カスタムサイズ。

型 **any**

IPdfSvgDrawSettings インターフェイス

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**
インターフェイス **IPdfImageDrawSettings**

SVG画像を描画するために**drawSvg** メソッドが使用する設定を表します。

プロパティ

- urlResolver

プロパティ

- urlResolver

相対URLを現在の要求パスに対して正しいURLに変換するために使用されるコールバック関数を決定します。 この関数には、引数として相対URLが渡され、解決されたURLを返す必要があります。

型 **Function**

IPdfTextDrawSettings インターフェイス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**
インターフェイス **IPdfTextSettings**

指定された**PdfPen** および**PdfBrush** を使用してテキストを描画するために、 **drawText** メソッドで 使用される設定を表します。

プロパティ

- brush
- font
- pen

プロパティ

- brush

テキストを塗りつぶすためのブラシを決定します。指定されていない場合は、デフォルトのドキュメント ブラシが使用されます（**setBrush** メソッド）。

型 **any**

- font

使用するフォントを決定します。指定されていない場合は、デフォルトのドキュメントフォントが 使用されます（ **setFont** メソッド）。

型 **PdfFont**

- pen

テキストをストロークするペンを決定します。指定されていない場合は、デフォルトのドキュメント ペンが使用されます（**setPen** メソッド）。

型 **any**

IPdfTextMeasurementInfo インターフェイス

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

measureText メソッドから返されるテキスト測定情報を表します。

プロパティ

- charCount
- size

プロパティ

- charCount

文字数を決定します。

型 **number**

- size

テキストサイズ（ポイント単位）を決定します。

型 **Size**

IPdfTextMeasurementSettings インターフェイス

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**
インターフェイス **IPdfTextSettings**

measureText メソッドによって使用される設定を表します。

プロパティ

- includeLastLineExternalLeading

プロパティ

- includeLastLineExternalLeading

最終行の外部先行値を測定結果に含めるかどうかを決定します。 デフォルト値はtrueです。

型 **boolean**

IPdfTextSettings インターフェイス

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

drawText および**measureText** メソッドで使用されるテキスト設定を表します。

プロパティ

- align
- characterSpacing
- columnGap
- columns
- continued
- ellipsis
- fill
- height
- indent
- lineBreak
- lineGap
- link
- paragraphGap
- strike
- stroke
- underline
- width
- wordSpacing

プロパティ

align

描画領域内にテキストを配置する方法を決定します。 デフォルト値は**Left**です。

型 **PdfTextHorizontalAlign**

characterSpacing

テキスト文字の間隔を決定します。 デフォルト値は0です。

型 **number**

columnGap

列の間隔（ポイント単位）を決定します。 デフォルト値は18です。

型 **number**

columns

テキストをフローする列の数を決定します。 デフォルト値は1です。

型 **number**

- continued

後続のテキストを直後に続けるか、新しい段落にするかを示します。trueの場合は、drawText呼び出し間でテキスト設定が 保持されます。すなわち、オプション引数は、前のdrawText呼び出しから 受け取った引数と結合されます。

デフォルト値はfalseです。

型 **boolean**

- ellipsis

テキストが指定された領域を超える場合に、テキストの末尾に表示される文字を決定します。 デフォルト値は未定義です。すなわち、省略記号は表示されません。 デフォルトの文字を使用する場合はtrueに設定します。

型 **any**

- fill

テキストを塗りつぶすかどうかを示します。 デフォルト値はtrueです。

型 **boolean**

- height

テキストがクリップされる描画領域の高さ（ポイント単位）を決定します。 デフォルト値は未定義です。すなわち、テキスト領域は 本文セクションの下端までに制限されます。 テキスト領域の高さを無限にするには、Infinityを使用します。

型 **number**

- indent

テキストの各段落のインデントの値（ポイント単位）を決定します。 デフォルト値は0です。

型 **number**

- lineBreak

行の折り返しを使用するかどうかを示します。 このプロパティは、**`width`** が定義されている場合は無視されます。 デフォルト値はtrueです。

型 **boolean**

- lineGap

テキストの行の間隔を決定します。 デフォルト値は0です。

型 **number**

- link

リンク注釈（URIアクション）を作成するために使用されるURLを決定します。

型 **string**

- paragraphGap

テキストの段落の間隔を決定します。 デフォルト値は0です。

型 **number**

- strike

テキストに取り消し線を付けるかどうかを示します。 デフォルト値はfalseです。

型 **boolean**

- stroke

テキストをストロークするかどうかを示します。 デフォルト値はfalseです。

型 **boolean**

- underline

テキストに下線を付けるかどうかを示します。 デフォルト値はfalseです。

型 **boolean**

- width

テキストが折り返されるテキスト領域の幅（ポイント単位）を決定します。 デフォルト値は未定義です。すなわち、テキスト領域は ページの右マージンまでに制限されます。 テキスト領域の幅を無限にするには、Infinityを使用します。 これが定義されている場合、**`lineBreak`** プロパティは強制的に有効になります。

型 **number**

- wordSpacing

テキストの単語の間隔を決定します。 デフォルト値は0です。

型 **number**

PdfFillRule 列挙体

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

点が閉じたパスの内部にあるかどうかを決定するルールを指定します。

メンバー

名前	値	説明
NonZero	0	非０ルール。
EvenOdd	1	偶数奇数ルール。

PdfImageHorizontalAlign 列挙体

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

画像の水平方向の配置を指定します。

メンバー

名前	値	説明
Left	0	画像を描画領域の左端に揃えます。
Center	1	画像を描画領域の中央に配置します。
Right	2	画像を描画領域の右端に揃えます。

PdfImageVerticalAlign 列挙体

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

画像の垂直方向の配置を指定します。

メンバー

名前	値	説明
Top	0	画像を描画領域の上端に揃えます。
Center	1	画像を描画領域の中央に配置します。
Bottom	2	画像を描画領域の下端に揃えます。

PdfLineCapStyle 列挙体

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

開いたサブパス（および破線）をストロークする際に、その終端に使用される形状を指定します。

メンバー

名前	値	説明
Butt	0	ストロークはパスの終点で四角になります。
Round	1	直径が線の幅と等しい半円孤が終点を中心に描画されて塗りつぶされます。
Square	2	ストロークは、パスの終点を線の幅の半分の距離だけ超えて四角になります。

PdfLineJoinStyle 列挙体

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

ストロークされるパスの角で使用される形状を指定します。

メンバー

名前	値	説明
Miter	0	２つのセグメントのストロークの外辺がそれらの交点まで延長されます。
Round	1	直径が線の幅と等しい円弧が２つのセグメントの交点を中心に描画されます。
Bevel	2	２つのセグメントはButtキャップで終了し、セグメントの終端より先にできたV字形の空間が三角形に塗りつぶされます。

PdfPageOrientation 列挙体

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

用紙の方向を指定します。

メンバー

名前	値	説明
Portrait	0	縦方向。
Landscape	1	横方向。

PdfPageSize 列挙体

ファイル `wijmo.pdf.js`
モジュール **wijmo.pdf**

ページサイズ（ポイント単位）を指定します。

メンバー

名前	値	説明
A0	0	A0ページサイズを表します。
A1	1	A1ページサイズを表します。
A2	2	A2ページサイズを表します。
A3	3	A3ページサイズを表します。
A4	4	A4ページサイズを表します。
A5	5	A5ページサイズを表します。
A6	6	A6ページサイズを表します。
A7	7	A7ページサイズを表します。
A8	8	A8ページサイズを表します。
A9	9	A9ページサイズを表します。
A10	10	A10ページサイズを表します。
B0	11	B0ページサイズを表します。
B1	12	B1ページサイズを表します。
B2	13	B2ページサイズを表します。
B3	14	B3ページサイズを表します。
B4	15	B4ページサイズを表します。
B5	16	B5ページサイズを表します。
B6	17	B6ページサイズを表します。
B7	18	B7ページサイズを表します。
B8	19	B8ページサイズを表します。
B9	20	B9ページサイズを表します。
B10	21	B10ページサイズを表します。
C0	22	C0ページサイズを表します。
C1	23	C1ページサイズを表します。
C2	24	C2ページサイズを表します。
C3	25	C3ページサイズを表します。
C4	26	C4ページサイズを表します。
C5	27	C5ページサイズを表します。
C6	28	C6ページサイズを表します。
C7	29	C7ページサイズを表します。
C8	30	C8ページサイズを表します。
C9	31	C9ページサイズを表します。
C10	32	C10ページサイズを表します。
RA0	33	RA0ページサイズを表します。
RA1	34	RA1ページサイズを表します。
RA2	35	RA2ページサイズを表します。
RA3	36	RA3ページサイズを表します。
RA4	37	RA4ページサイズを表します。
SRA0	38	SRA0ページサイズを表します。
SRA1	39	SRA1ページサイズを表します。
SRA2	40	SRA2ページサイズを表します。
SRA3	41	SRA3ページサイズを表します。
SRA4	42	SRA4ページサイズを表します。
Executive	43	Executiveページサイズを表します。
Folio	44	Folioページサイズを表します。
Legal	45	Legalページサイズを表します。
Letter	46	Letterページサイズを表します。
Tabloid	47	Tabloidページサイズを表します。

PdfTextHorizontalAlign 列挙体

ファイル wijmo.pdf.js
モジュール **wijmo.pdf**

テキストコンテンツの水平方向の配置を指定します。

メンバー

名前	値	説明
Left	0	左詰め。
Center	1	テキストは中央に揃えられます。
Right	2	テキストは右に揃えられます。
Justify	3	両端揃え。

wijmo.grid.pdf モジュール

ファイル `wijmo.grid.pdf.js`
モジュール **wijmo.grid.pdf**

FlexGrid をPDFにエクスポートするために使用される**FlexGridPdfConverter** クラスを定義します。

クラス

- FlexGridPdfConverter
- PdfFormatItemEventArgs
- PdfWebWorker
- PdfWebWorkerClient
- PdfWebWorkerExportDoneEventArgs

インターフェイス

- ICellStyle
- IClientData
- IClientDataItem
- IFlexGridDrawSettings
- IFlexGridExportSettings
- IFlexGridStyle

列挙体

- ExportMode
- ScaleMode

FlexGridPdfConverter クラス

ファイル `wijmo.grid.pdf.js`
モジュール **wijmo.grid.pdf**

FlexGrid をPDFにエクスポートする機能を提供します。

次の例では、**FlexGridPdfConverter** を使用して**FlexGrid** をPDFにエクスポートする方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/equxqkyt)

メソッド

- draw
- drawToPosition
- export

メソッド

STATIC draw

```
draw(flex: any, doc: PdfDocument, width?: number, height?: number, settings?: IFlexGridDrawSettings): void
```

FlexGrid を既存の**PdfDocument** の座標 (0, @wijmo.pdf.PdfDocument.y)に描画します。

幅が指定されていない場合、グリッドは実際のサイズでレンダリングされ、 必要に応じて改ページされます。高さが指定されていない場合、グリッドは 幅に合わせて拡大縮小され、必要に応じて垂直に改ページされます。 幅と高さが両方とも決定されている場合、グリッドは改ページなしで、 指定された四角形に合わせて拡大縮小されます。

```
var doc = new wijmo.pdf.PdfDocument({
   ended: function (sender, args) {
      wijmo.pdf.saveBlob(args.blob, 'FlexGrid.pdf');
   }
});

wijmo.grid.pdf.FlexGridPdfConverter.draw(grid, doc, null, null, {
   maxPages: 10,
   styles: {
      cellStyle: {
         backgroundColor: '#ffffff',
         borderColor: '#c6c6c6'
      },
      headerCellStyle: {
         backgroundColor: '#eaeaea'
      }
   }
});
```

パラメーター

- **flex**: **any**
  エクスポートする**FlexGrid** インスタンス。
- **doc**: **PdfDocument**
  描画先の**PdfDocument** インスタンス。
- **width**: **number** OPTIONAL
  描画領域の幅（ポイント単位）。
- **height**: **number** OPTIONAL
  描画領域の高さ（ポイント単位）。
- **settings**: **IFlexGridDrawSettings** OPTIONAL
  描画の設定。

戻り値 **void**

STATIC drawToPosition

```
drawToPosition(flex: any, doc: PdfDocument, point: Point, width?: number, height?: number, settings?: IFlexGridDrawSettings): void
```

FlexGrid を既存の**PdfDocument** インスタンスの指定された座標に 描画します。

幅が指定されていない場合、グリッドは改ページなしで、実際のサイズで レンダリングされます。 高さが指定されていない場合、グリッドは改ページなしで、幅に合わせてスケーリングされて レンダリングされます。 幅と高さが両方とも決定されている場合、グリッドは改ページなしで、 指定された四角形に合わせて拡大縮小されます。

```
var doc = new wijmo.pdf.PdfDocument({
   ended: function (sender, args) {
      wijmo.pdf.saveBlob(args.blob, 'FlexGrid.pdf');
   }
});

wijmo.grid.pdf.FlexGridPdfConverter.drawToPosition(grid, doc, new wijmo.Point(0, 0), null, null, {
   maxPages: 10,
   styles: {
      cellStyle: {
         backgroundColor: '#ffffff',
         borderColor: '#c6c6c6'
      },
      headerCellStyle: {
         backgroundColor: '#eaeaea'
      }
   }
});
```

パラメーター

- **flex**: **any**
  エクスポートする**FlexGrid** インスタンス。
- **doc**: **PdfDocument**
  描画先の**PdfDocument** インスタンス。
- **point**: **Point**
  描画する位置（ポイント単位）。
- **width**: **number** OPTIONAL
  描画領域の幅（ポイント単位）。
- **height**: **number** OPTIONAL
  描画領域の高さ（ポイント単位）。
- **settings**: **IFlexGridDrawSettings** OPTIONAL
  描画の設定。

戻り値 **void**

STATIC export

```
export(flex: FlexGrid, fileName: string, settings?: IFlexGridExportSettings): void
```

FlexGrid をPDFにエクスポートします。

```
wijmo.grid.pdf.FlexGridPdfConverter.export(grid, 'FlexGrid.pdf', {
   scaleMode: wijmo.grid.pdf.ScaleMode.PageWidth,
   maxPages: 10,
   styles: {
      cellStyle: {
         backgroundColor: '#ffffff',
         borderColor: '#c6c6c6'
      },
      headerCellStyle: {
         backgroundColor: '#eaeaea'
      }
   },
   documentOptions: {
      info: {
         title: 'Sample'
      }
   }
});
```

パラメーター

- **flex**: **FlexGrid**
  エクスポートする**FlexGrid** インスタンス。

- **fileName**: **string**
  エクスポートするファイルの名前。

- **settings**: **IFlexGridExportSettings** OPTIONAL
  エクスポートの設定。

戻り値 **void**

PdfFormatItemEventArgs クラス

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**

IFlexGridDrawSettings.formatItemコールバックの引数を表します。

コンストラクタ

- constructor

プロパティ

- cancelBorders
- canvas
- cell
- clientRect
- col
- contentRect
- data
- panel
- range
- row
- style
- textTop

メソッド

- getFormattedCell

コンストラクタ

constructor

```
constructor(p: any /*_IGridPanel*/, rng: any /*_ICellRange*/, cell: HTMLElement, canvas: PdfPageArea, clientRect: Rect, contentRect: Rect, textTop: number, style: ICellStyle, getFormattedCell?: Function): PdfFormatItemEventArgs
```

PdfFormatItemEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **p**: **any /*_IGridPanel*/**
  範囲を含む**GridPanel**。
- **rng**: **any /*_ICellRange*/**
  イベントの影響を受けるセルの範囲。
- **cell**: **HTMLElement**
  レンダリングされるグリッドセルを表す要素。
- **canvas**: **PdfPageArea**
  カスタム描画を実行するキャンバス。
- **clientRect**: **Rect**
  レンダリングされるグリッドセルのクライアント四角形をキャンバス座標で表すオブジェクト。
- **contentRect**: **Rect**
  レンダリングされるグリッドセルのコンテンツ四角形をキャンバス座標で表すオブジェクト。
- **textTop**: **number**
  テキストの上位置をキャンバス座標で表すオブジェクト。
- **style**: **ICellStyle**
  レンダリングされるグリッドセルのスタイルを表すオブジェクト。
- **getFormattedCell**: **Function** OPTIONAL
  getFormattedCellメソッドが呼び出されたときにグリッドセルを返すコールバック関数。

戻り値 **PdfFormatItemEventArgs**

プロパティ

cancelBorders

デフォルトのセル境界線の描画をキャンセルすることを示す値を取得または設定します。

型 **boolean**

canvas

カスタム描画を実行するキャンバスを取得します。

型 **PdfPageArea**

cell

レンダリングされているグリッドセルを表す要素への参照を取得します。 IFlexGridDrawSettings.customCellContentがtrueに設定されている場合は、書式設定されたグリッドセルを表す要素への参照を含みます。そうでない場合は、null値です。

型 **HTMLElement**

- clientRect

レンダリングされているセルのクライアント四角形をキャンバス座標で取得します。

型 **Rect**

- col

Gets the column affected by this event.

型 **number**

- contentRect

レンダリングされているセルのコンテンツ四角形をキャンバス座標で取得します。

型 **Rect**

- data

Gets or sets the data associated with the event.

型 **any**

panel

Gets the **GridPanel** affected by this event.

型 **GridPanel**

- range

Gets the **CellRange** affected by this event.

型 **CellRange**

- row

Gets the row affected by this event.

型 **number**

- style

レンダリングされているセルのスタイルを表すオブジェクトを取得します。 IFlexGridDrawSettings.customCellContentがtrueに設定されている場合、スタイルは セルスタイルに基づいて決定されます。そうでない場合は、エクスポートされるセルの行タイプに基づいて、 IFlexGridDrawSettings.styles エクスポート設定の組み合わせが含まれます。

型 **ICellStyle**

- textTop

レンダリングされているセルのテキストの上位置をキャンバス座標で表す値を取得します。

型 **number**

メソッド

getFormattedCell

```
getFormattedCell(): HTMLElement
```

レンダリングされているグリッドセルを表す要素への参照を返します。 このメソッドは、カスタム書式のエクスポートが無効だが、いくつかのセルでカスタムコンテンツを エクスポートする必要がある場合に便利です。

戻り値 **HTMLElement**

PdfWebWorker クラス

ファイル　wijmo.grid.pdf.js
モジュール　**wijmo.grid.pdf**

Represents server-side methods for exporting FlexGrid to PDF/generating PDF, for use with Web Worker.

メソッド

メソッド

STATIC deserializeGrid

```
deserializeGrid(data: ArrayBuffer, settings?: IFlexGridDrawSettings): any
```

Deserializes the **FlexGrid** from ArrayBuffer to its internal representation that can be used in a Web Worker and passed to the **draw** and **drawToPosition** methods.

パラメーター

- **data**: **ArrayBuffer**
  The data to deserialize.
- **settings**: **IFlexGridDrawSettings** OPTIONAL
  The draw settings used to deserialize the grid.

戻り値　**any**

STATIC initExport

```
initExport(draw: Function): void
```

Performs the PDF document generation started in a UI thread by calling the **export** method.

パラメーター

- **draw**: **Function**
  The callback function to draw PDF. The function takes two parameters:
  - **doc**: An instance of the **PdfDocument** class.
  - **clientData**: A dictionary of {name: value} pairs that contains the data added on the client side.

戻り値　**void**

STATIC initExportGrid

```
initExportGrid(): void
```

Performs the export started in a UI thread by calling the **exportGrid** method.

戻り値　**void**

STATIC sendExportProgress

```
sendExportProgress(value: number): void
```

Sends the progress value to a client, where it will be handled by the **`export`**'s progress callback function. Should be used in conjunction with the **`export`** method to inform client about the progress of the export.

パラメーター

- **value**: **number**
  The progress value, in the range of [0.0..1.0].

戻り値 **void**

PdfWebWorkerClient クラス

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**

Represents client-side methods for exporting FlexGrid to PDF/generating PDF, for use with Web Worker.

メソッド

メソッド

- STATIC addGrid

```
addGrid(worker: Worker, grid: FlexGrid, name: string, settings: IFlexGridDrawSettings): void
```

Adds named FlexGrid with settings, which will be used in a Web Worker to generate a PDF document. This method should be used in conjunction with the **`export`** method.

パラメーター

- **worker**: **Worker**
  The Web Worker instance to send the data to.
- **grid**: **FlexGrid**
  The grid
- **name**: **string**
  The name associated with the grid.
- **settings**: **IFlexGridDrawSettings**
  The draw settings.

戻り値 **void**

STATIC addImage

```
addImage(worker: Worker, image: string, name: string, settings: IPdfImageDrawSettings): void
```

Adds named image with settings, which will be used in a Web Worker to generate a PDF document. This method should be used in conjunction with the **`export`** method.

パラメーター

- **worker**: **Worker**
  The Web Worker instance to send the data to.
- **image**: **string**
  A string containing the URL to get the image from or the data URI containing a base64 encoded image.
- **name**: **string**
  The name associated with the image.
- **settings**: **IPdfImageDrawSettings**
  The image drawing settings.

戻り値 **void**

STATIC addString

```
addString(worker: Worker, value: string, name: string): void
```

Adds named string which will be used in a Web Worker code to generate a PDF document. This method should be used in conjunction with the **`export`** method.

パラメーター

- **worker**: **Worker**
  The Web Worker instance to send the data to.
- **value**: **string**
  The value.
- **name**: **string**
  The name associated with the string.

戻り値 **void**

STATIC export

```
export(worker: Worker, settings: any, done: Function, progress?: Function): void
```

Exports PDF in a background thread.

パラメーター

- **worker**: **Worker**
  The Web Worker instance to run the exporting code in.
- **settings**: **any**
  An object containing **`PdfDocument`**'s initialization settings.
- **done**: **Function**
  The callback function to call when drawing is done. The function takes a single parameter, an instance of the **`PdfWebWorkerExportDoneEventArgs`** class.
- **progress**: **Function** OPTIONAL
  An optional function that gives feedback about the progress of the export. The function takes a single parameter, a number changing from 0.0 to 1.0, where the value of 0.0 indicates that the operation has just begun and the value of 1.0 indicates that the operation has completed.

戻り値 **void**

STATIC exportGrid

```
exportGrid(worker: Worker, grid: FlexGrid, fileName: string, settings: IFlexGridExportSettings, done?: Function, progress?: Function): void
```

Exports the **`FlexGrid`** to PDF in a background thread.

パラメーター

- **worker**: **Worker**
  The Web Worker instance to run the exporting code in.
- **grid**: **FlexGrid**
  The **`FlexGrid`** instance to export.
- **fileName**: **string**
  The name of the file to export.
- **settings**: **IFlexGridExportSettings**
  The export settings.
- **done**: **Function** OPTIONAL
  An optional callback function to call when exporting is done. The function takes a single parameter, an instance of the **`PdfWebWorkerExportDoneEventArgs`** class.
- **progress**: **Function** OPTIONAL
  An optional function that gives feedback about the progress of the export. The function takes a single parameter, a number changing from 0.0 to 1.0, where the value of 0.0 indicates that the operation has just begun and the value of 1.0 indicates that the operation has completed.

戻り値 **void**

STATIC serializeGrid

```
serializeGrid(grid: FlexGrid, settings?: IFlexGridExportSettings): ArrayBuffer
```

Serializes the **FlexGrid** to ArrayBuffer. The serialized data can be send to a Web Worker using the postMessage method.

パラメーター

- **grid**: **FlexGrid**
  The **FlexGrid** instance to serialize.

- **settings**: **IFlexGridExportSettings** OPTIONAL
  The export settings used to serialize the grid.

戻り値 **ArrayBuffer**

PdfWebWorkerExportDoneEventArgs クラス

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

Provides arguments for the callback parameter of the **exportGrid** and **export** methods.

コンストラクタ

- constructor

プロパティ

- blob
- buffer
- empty

コンストラクタ

constructor

```
constructor(buffer: ArrayBuffer): PdfWebWorkerExportDoneEventArgs
```

Initializes a new instance of the **PdfWebWorkerExportDoneEventArgs** class.

パラメーター

- **buffer**: **ArrayBuffer**
  An ArrayBuffer.

戻り値 **PdfWebWorkerExportDoneEventArgs**

プロパティ

- blob

Gets a Blob object that contains the document data.

型 **Blob**

- buffer

Gets an ArrayBuffer that contains the document data.

型 **ArrayBuffer**

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

ICellStyle インターフェイス

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**

セルのルックアンドフィールを表します。

プロパティ

- backgroundColor
- borderColor
- font

プロパティ

backgroundColor

セルの背景色を表します。

型 **string**

borderColor

セルの境界線の色を表します。

型 **string**

font

セルのフォントを表します。

型 **any**

IClientData インターフェイス

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**

Represents a dictionary of {name: value} pairs which contains client data.

IClientDataItem インターフェイス

ファイル `wijmo.grid.pdf.js`
モジュール **wijmo.grid.pdf**

Represents a single item of the **IClientData** dictionary.

プロパティ

- content
- settings

プロパティ

- content

Gets or sets the content for the data item.

型 **any**

- settings

Gets or sets the settings for the data item.

型 **any**

IFlexGridDrawSettings インターフェイス

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**

draw および **drawToPosition** メソッドによって使用される設定を表します。

プロパティ

- customCellContent
- drawDetailRows
- embeddedFonts
- exportMode
- formatItem
- maxPages
- progress
- recalculateStarWidths
- repeatMergedValuesAcrossPages
- styles

プロパティ

customCellContent

カスタムセルの内容とスタイルを評価およびエクスポートするかどうかを示します。 trueに設定すると、エクスポートロジックにより、cell.textContentプロパティを使用してセルの内容が取得され、getComputedStyle(cell)を使用してセルのスタイルが取得されます。 デフォルトは'undefined'（false）です。

型 **boolean**

drawDetailRows

詳細行を描画するかどうかを示します。 このプロパティがfalseの場合、詳細行は無視されます。それ以外の場合は、詳細行は空になり、内容をformatItemイベントハンドラを使用して手動で描画する必要があります。 デフォルトは'undefined'（false）です。

型 **boolean**

embeddedFonts

ドキュメントに埋め込まれる独自フォントの配列を表します。

次のサンプルは、FlexGridPdfConverterを設定して、2つの独自フォント（Cuprum-Bold.ttfとCuprum-Regular.ttf）を使用する方法を示しています。最初のフォントはヘッダーセルだけに 適用され、2番目のフォントは残りのすべてのセルに適用されます。

```
wijmo.grid.pdf.FlexGridPdfConverter.export(flex, fileName, {
   embeddedFonts: [{
      source: 'resources/ttf/Cuprum-Bold.ttf',
      name: 'cuprum',
      style: 'normal',
      weight: 'bold'
   }, {
      source: 'resources/ttf/Cuprum-Regular.ttf',
      name: 'cuprum',
      style: 'normal',
      weight: 'normal'
   }],
   styles: {
      cellStyle: {
         font: {
            family: 'cuprum'
         }
      },
      headerCellStyle: {
         font: {
            weight: 'bold'
         }
      }
   }
});
```

型　　**IPdfFontFile**[]

exportMode

エクスポートモードを決定します。

型　　**ExportMode**

formatItem

エクスポートされたセルごとに呼び出されるオプションのコールバック関数。エクスポートされたセルの値およびスタイルの 変換やカスタム描画を行うことができます。

この関数は、最初の引数として**PdfFormatItemEventArgs** クラスインスタンスを受け取ります。

カスタム描画の場合は、**cancel** プロパティをtrueに設定してデフォルトのセルコンテンツの描画をキャンセルし、 **cancelBorders** プロパティをtrueに設定してデフォルトのセル境界線の描画をキャンセルします。

```
wijmo.grid.pdf.FlexGridPdfConverter.export(flex, fileName, {
   formatItem: function(args) {

       // "Country"列の通常のセルの背景色を変更します。
       if (args.panel.cellType === wijmo.grid.CellType.Cell && args.panel.columns[args.col].binding === "country") {
           args.style.backgroundColor = 'blue';
       }
   }
});
```

型　　**Function**

- maxPages

エクスポートする最大ページ数を決定します。

型 **number**

- progress

An optional function that gives feedback about the progress of a task. The function accepts a single argument, a number changing from 0.0 to 1.0, where the value of 0.0 indicates that the operation has just begun and the value of 1.0 indicates that the operation has completed.

```
wijmo.grid.pdf.FlexGridPdfConverter.export(flex, fileName, {
   progress: function(value) {

       // Handle the progress here.
   }
});
```

型 **Function**

- recalculateStarWidths

グリッドの幅ではなくPDFページ幅を使用してスターサイズの列の幅を再計算するかどうかを示します。

型 **boolean**

- repeatMergedValuesAcrossPages

結合範囲が複数のページに分かれた場合に、結合値を複数のページに繰り返し表示するかどうかを示します。

型 **boolean**

- styles

エクスポートされた**FlexGrid** のルックアンドフィールを表します。

型 **IFlexGridStyle**

IFlexGridExportSettings インターフェイス

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**
インターフェイス **IFlexGridDrawSettings**

export メソッドによって使用される設定を表します。

プロパティ

- documentOptions
- scaleMode

プロパティ

documentOptions

基底の**PdfDocument** のオプションを表します。

型 **any**

scaleMode

拡大縮小モードを決定します。

型 **ScaleMode**

IFlexGridStyle インターフェイス

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**

エクスポートされる**FlexGrid** のルックアンドフィールを表します。

プロパティ

- altCellStyle
- cellStyle
- errorCellStyle
- footerCellStyle
- groupCellStyle
- headerCellStyle

プロパティ

- altCellStyle

FlexGrid の奇数番号の行に適用されるセルスタイルを表します。

型 **ICellStyle**

- cellStyle

FlexGrid 内のセルに適用されるセルスタイルを指定します。

型 **ICellStyle**

- errorCellStyle

showErrors プロパティが有効な場合、検証エラーが含まれるFlexGridのセルに適用されたセルスタイルを表します。

型 **ICellStyle**

- footerCellStyle

FlexGrid の列フッターに適用されるセルスタイルを表します。

型 **ICellStyle**

- groupCellStyle

FlexGrid のグループ化された行に適用されるセルスタイルを表します。

型 **ICellStyle**

- headerCellStyle

FlexGrid の行ヘッダーと列ヘッダーに適用されるセルスタイルを 表します。

型 **ICellStyle**

ExportMode 列挙体

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**

グリッド全体と 1 つのセクションのどちらをレンダリングするかを指定します。

メンバー

名前	値	説明
All	0	グリッドからすべてのデータをエクスポートします。
Selection	1	現在の選択内容だけをエクスポートします。

ScaleMode 列挙体

ファイル wijmo.grid.pdf.js
モジュール **wijmo.grid.pdf**

ページに合わせてグリッドコンテンツを拡大縮小する方法を指定します。

メンバー

名前	値	説明
ActualSize	0	グリッドを実際のサイズでレンダリングし、必要に応じて改ページを行います。
PageWidth	1	ページ幅に合わせてグリッドを拡大縮小します。
SinglePage	2	1 ページに合わせてグリッドを拡大縮小します。

wijmo.nav モジュール

ファイル `wijmo.nav.js`
モジュール **wijmo.nav**

TabPanel 、**TreeView** と関連クラスを含むナビゲーションコントロールを定義します。

クラス

- FormatNodeEventArgs
- Tab
- TabPanel
- TreeNode
- TreeNodeDragDropEventArgs
- TreeNodeEventArgs
- TreeView

列挙体

- DropPosition

FormatNodeEventArgs クラス

ファイル wijmo.nav.js
モジュール **wijmo.nav**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

formatItem イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- dataItem
- element
- empty
- level

コンストラクタ

constructor

```
constructor(dataItem: any, element: HTMLElement, level: number): FormatNodeEventArgs
```

FormatNodeEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **dataItem**: **any**
  このノードで表されるデータ項目。
- **element**: **HTMLElement**
  書式設定されるノードを表す要素。
- **level**: **number**
  書式設定されるノードのアウトラインレベル。

戻り値 **FormatNodeEventArgs**

プロパティ

dataItem

書式設定されるデータ項目を取得します。

型 **any**

element

書式設定されるノードを表す要素への参照を取得します。

型 **HTMLElement**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

level

書式設定されるノードのアウトラインレベルを取得します。

型 **number**

Tab クラス

ファイル wijmo.nav.js
モジュール **wijmo.nav**
派生クラス **WjTab**

TabPanel コントロール内のタブを表します。

タブには、ヘッダーとペインの2つの要素があります。ヘッダーにはタブのタイトルが表示され、ペインにはタブの内容が表示されます。

コンストラクタ

- constructor

プロパティ

- header
- isDisabled
- isVisible
- pane
- tabPanel

コンストラクタ

constructor

```
constructor(header: any, pane: any): Tab
```

Tab クラスの新しいインスタンスを初期化します。

パラメーター

- **header**: **any**
  タブヘッダーを含む要素または、要素に該当するCSSセレクター。
- **pane**: **any**
  タブの内容を含む要素または要素に該当するCSSセレクター。

戻り値 **Tab**

プロパティ

- header

タブのヘッダ要素を取得します。

型 **HTMLElement**

- isDisabled

この **Tab** が無効かどうかを示す値を取得または設定します。

型 **boolean**

- isVisible

このTab が表示されるかどうかを示す値を取得または設定します。

型 **boolean**

pane

タブの内容要素を取得します。

型 **HTMLElement**

- tabPanel

このタブを含むTabPanel への参照を取得します。

型 **TabPanel**

TabPanel クラス

ファイル wijmo.nav.js
モジュール **wijmo.nav**
基本クラス **Control**
派生クラス **WjTabPanel**
表示 ☐継承されたメンバー ☐イベント発生元

TabPanel を使用すると、ビュー、データセット、またはアプリケーションの機能面を切り替えるなどのような高度なレベルでコンテンツを整理できます。

タブは、関連する内容の上側に1行で表示されます。 タブヘッダーは、タブ内の内容を簡潔に記述します。

タブは、マウスまたはキーボードで選択でき、現在の選択内容を反映するように内容を自動的に更新することができます。

次の例では、**TabPanel** を使用して内容をページに整理する方法を示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/17tmuomr)

コンストラクタ

- constructor

プロパティ

- autoSwitch
- controlTemplate
- hostElement
- isAnimated
- isDisabled
- isTouching
- isUpdating
- rightToLeft
- selectedIndex
- selectedTab
- tabs

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTab
- getTemplate
- initialize
- invalidate
- invalidateAll

- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged

コンストラクタ

constructor

```
constructor(element: any, options?, keepChildren?: boolean): TabPanel
```

TabPanel クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。
- **keepChildren**: **boolean** OPTIONAL
  子要素を保持するかどうか。 trueに設定した場合、呼び出し元はDOMに基づいて**tabs** の配列に値を設定します。

戻り値 **TabPanel**

プロパティ

- autoSwitch

ユーザーが矢印キーを使用してタブを選択したときにコントロールがタブを自動的に切り替えるかどうかを決定する値を取得または設定します。

autoSwitch がtrue（デフォルト値）に設定されている場合、矢印キーを押すとタブが自動的に切り替わります。タブキーを押すと、選択されていないタブヘッダーは除外され、タブ順序の次の要素が選択されます。

autoSwitch がfalseに設定されている場合、矢印キーまたはTabキーを押すと、フォーカスが次のタブヘッダーまたは前のタブヘッダーに移動しますが、タブは切り替わりません。フォーカスがあるタブをアクティブにするには、EnterキーまたはSpaceキーを押す必要があります。

ほとんどの場合、デフォルト値は適切な（アクセス可能な）動作を提供しますが、 **autoSwitch** をfalseに設定したいユーザーも存在するかもしれません。 このトピックの詳細については、以下をご参考ください。 **W3C ARIA practices** と **SimplyAccessible** articles.

型 **boolean**

STATIC controlTemplate

TabPanel コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isAnimated

タブの変更をフェードイン効果でアニメーション化するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectedIndex

現在選択されている（アクティブな）タブのインデックスを取得または設定します。

型 **number**

selectedTab

現在選択されている**Tab** を取得または設定します。

型 **Tab**

tabs

header と**pane** プロパティが**TabPanel** コントロールの内容を決定する**Tab** オブジェクトの配列を取得します。

型 **ObservableArray**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTab

```
getTab(id: string): Tab
```

IDまたはヘッダの内容によって**Tab** を取得します。

パラメーター

- **id**: **string**
  取得する**TabのID**。

戻り値 **Tab**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

引数 **EventArgs**

TreeNode クラス

ファイル wijmo.nav.js
モジュール **wijmo.nav**

TreeView のノードを表すクラス。

コンストラクタ

- constructor

プロパティ

- checkBox
- dataItem
- element
- hasChildren
- hasPendingChildren
- index
- isChecked
- isCollapsed
- isDisabled
- itemsSource
- level
- nodes
- parentNode
- treeView

メソッド

- addChildNode
- ensureVisible
- equals
- move
- next
- nextSibling
- previous
- previousSibling
- refresh
- remove
- select
- setChecked
- setCollapsed

コンストラクタ

constructor

```
constructor(treeView: TreeView, nodeElement: HTMLElement): TreeNode
```

TreeNode の新しいインスタンスを初期化します。

パラメーター

- **treeView**: **TreeView**
  このノードが含まれる**TreeView**。

- **nodeElement**: **HTMLElement**
  TreeView のこのノードを表すHTML要素。

戻り値 **TreeNode**

プロパティ

checkBox

このノードに関連付けられたチェックボックスを表すHTMLInputElementを取得します。

型 **HTMLInputElement**

dataItem

このノードが表すデータ項目を取得します。

型 **any**

element

TreeView のこのノードを表すHTML要素を取得します。

型 **HTMLElement**

hasChildren

このノードが子ノードを持つかどうかを示す値を取得します。

型 **boolean**

hasPendingChildren

このノードが展開されたときに、遅延ロードされる保留中の子ノードを持つか どうかを示す値を取得します。

型 **boolean**

index

親のノードコレクション内でのこのノードのインデックスを取得します。

型 **number**

- isChecked

このノードがオンかどうかを示す値を取得または設定します。

このプロパティの値が変化すると、子ノードと祖先ノードが自動的に更新され、 親の**TreeView** の**checkedItemsChanged** イベントが 発生します

型 **boolean**

- isCollapsed

このノードが展開されているか、折りたたまれているかを示す値を取得または設定します。

型 **boolean**

- isDisabled

このノードが無効かどうかを示す値を取得または設定します。

無効化されたノードは、マウスイベントやキーボードイベントを取得できません。

型 **boolean**

- itemsSource

この**TreeNode** の項目を含む配列を取得します。

このプロパティは読み取り専用です。親**TreeView** の**itemsSource** 配列のメンバである配列を返します。

型 **any**[]

- level

このノードのレベル。

最上位ノードのレベルは0です。

型 **number**

- nodes

このノードの子ノードを含む配列を取得します。

このノードが子を持たない場合、このプロパティはnullを返します。

型 **TreeNode**[]

- parentNode

ノードの親ノードを取得します。

最上位ノードでは、このプロパティはnullを返します。

型 **TreeNode**

treeView

このノードを含む**TreeView** への参照を取得します。

型 **TreeView**

メソッド

addChildNode

```
addChildNode(index: number, dataItem: any): TreeNode
```

特定の位置に子ノードを追加します。

パラメーター

- **index**: **number**
  新しい子ノードのインデックス。
- **dataItem**: **any**
  新しいノードの作成に使用されたデータ項目。

戻り値 **TreeNode**

ensureVisible

```
ensureVisible(): void
```

折りたたまれた祖先ノードがあれば展開し、要素をビュー内にスクロールして、 ノードが表示されるようにします。

戻り値 **void**

equals

```
equals(node: TreeNode): boolean
```

このノードが別のノードと同じ要素を参照するかどうかを確認します。

パラメーター

- **node**: **TreeNode**
  このオブジェクトと比較する@TreeNode。

戻り値 **boolean**

move

```
move(refNode: any, position: DropPosition): boolean
```

この**TreeNode** を**TreeView** の新しい位置に移動します。

パラメーター

- **refNode**: **any**
  ノードが移動される位置を定義する参照**TreeNode**。

- **position**: **DropPosition**
  ノードの移動先を参照ノードの前、後、中のどこにするか。

戻り値 **boolean**

next

```
next(visible?: boolean, enabled?: boolean): TreeNode
```

ビュー内の次のノードへの参照を取得します。

パラメーター

- **visible**: **boolean** OPTIONAL
  可視のノード（祖先ノードが折りたたまれていない）だけを返すかどうか。

- **enabled**: **boolean** OPTIONAL
  有効なノード（祖先ノードが無効でない）だけを返すかどうか。

戻り値 **TreeNode**

nextSibling

```
nextSibling(): TreeNode
```

ビュー内の次の兄弟ノードへの参照を取得します。

戻り値 **TreeNode**

previous

```
previous(visible?: boolean, enabled?: boolean): TreeNode
```

ビュー内の前のノードへの参照を取得します。

パラメーター

- **visible**: **boolean** OPTIONAL
  可視のノード（祖先ノードが折りたたまれていない）だけを返すかどうか。

- **enabled**: **boolean** OPTIONAL
  有効なノード（祖先ノードが無効でない）だけを返すかどうか。

戻り値 **TreeNode**

previousSibling

```
previousSibling(): TreeNode
```

ビュー内の前の兄弟ノードへの参照を取得します。

戻り値 **TreeNode**

refresh

```
refresh(dataItem?: any): void
```

データ変更を反映するためにノードを更新します。

パラメーター

- **dataItem**: **any** OPTIONAL
  新しいノードのデータ。 このパラメータはオプションです。 指定されていない場合は、元のデータ項目（おそらく更新されている）に基づいてノードが更新されます。

戻り値 **void**

remove

```
remove(): void
```

この**TreeNode** を**TreeView** から削除します。

戻り値 **void**

select

```
select(): void
```

このノードを選択します。

戻り値 **void**

setChecked

```
setChecked(checked: boolean, updateParent?: boolean): void
```

このノードと子をオンの状態に設定します。

パラメーター

- **checked**: **boolean**
  ノードと子をオンにするかオフにするか。

- **updateParent**: **boolean** OPTIONAL
  このノードの祖先ノードのオン状態を更新するかどうか。

戻り値 **void**

setCollapsed

```
setCollapsed(collapsed: boolean, animate?: boolean, collapseSiblings?: boolean): void
```

ノードを折りたたまれた状態に設定します。

パラメーター

- **collapsed**: **boolean**
  ノードを折りたたむか、展開するか。
- **animate**: **boolean** OPTIONAL
  新しい状態を適用するときに、アニメーションを使用するかどうか。
- **collapseSiblings**: **boolean** OPTIONAL
  このノードを展開するときに、兄弟ノードを折りたたむかどうか。

戻り値 **void**

TreeNodeDragDropEventArgs クラス

ファイル wijmo.nav.js
モジュール **wijmo.nav**
基本クラス **CancelEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

TreeNode ドラッグアンドドロップイベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- dragSource
- dropTarget
- empty
- position

コンストラクタ

constructor

```
constructor(dragSource: TreeNode, dropTarget: TreeNode, position: DropPosition): TreeNodeDragDropEventArgs
```

TreeNodeEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **dragSource**: **TreeNode**
  ドラッグされる**TreeNode**。
- **dropTarget**: **TreeNode**
  ドラッグされるノードがドロップされる**TreeNode**。
- **position**: **DropPosition**
  このイベントが参照する**DropPosition**。

戻り値 **TreeNodeDragDropEventArgs**

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

- dragSource

ドラッグされる**TreeNode** への参照を取得します。

型 **TreeNode**

dropTarget

現在の**TreeNode** ターゲットへの参照を取得します。

型 **TreeNode**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

position

TreeNode をドロップする場所を指定する**DropPosition** 値を 取得または設定します。

型 **DropPosition**

TreeNodeEventArgs クラス

ファイル wijmo.nav.js
モジュール **wijmo.nav**
基本クラス **CancelEventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

TreeNode 関連イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- cancel
- empty
- node

コンストラクタ

constructor

```
constructor(node: TreeNode): TreeNodeEventArgs
```

TreeNodeEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **node**: **TreeNode**
  このイベントが参照する**TreeNode**。

戻り値 **TreeNodeEventArgs**

プロパティ

- cancel

イベントをキャンセルするかどうかを示す値を取得または設定します。

継承元 **CancelEventArgs**
型 **boolean**

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

- node

このイベントが参照する**TreeNode** を取得します。

型 **TreeNode**

TreeView クラス

ファイル wijmo.nav.js
モジュール **wijmo.nav**
基本クラス **Control**
派生クラス **WjTreeView**
表示 ☐継承されたメンバー ☐イベント発生元

TreeView コントロールは、テキスト、チェックボックス、画像、 または任意のHTMLコンテンツを保持できる**TreeNode** オブジェクトの階層リストを表示します。

通常、**TreeView** は、階層として表示すると便利なドキュメントの見出し、 インデックス項目、ディスク内のファイルやディレクトリなどの 情報を表示するために使用されます。

TreeView を作成したら、通常は次のプロパティを設定します。

1. **itemsSource**：ツリーに表示されるデータを含む配列。
2. **displayMemberPath**：ノードに表示するテキストを含む データ項目プロパティの名前（デフォルトは'header'）。
3. **childItemsPath**：ノードの子項目を含むデータ項目プロパティの 名前（デフォルトは'items'）。

TreeView コントロールは、次のキーボードコマンドをサポートしています。

キーの組み合わせアクション
↑/↓ 表示されている前のノードまたは次のノードを選択します。
← 選択したノードに子ノードが存在する場合はノードを折りたたむ、それ以外の場合は親ノードを選択します。
→ 選択したノードに子ノードが存在する場合は、ノードを展開します。
Home/End 表示されている最初または最後の可視ノードを選択します。
Space 現在のノード内のチェックボックスを切り替えます(**showCheckboxes** プロパティを参照)。
その他の文字 入力されたテキスト（複数文字の自動検索）を含むノードを検索します。

次の例では、単純なツリーを作成し、TreeViewの主なプロパティの効果を確認することができます。

サンプル

Show me (https://jsfiddle.net/Wijmo5/egmg93wc)

コンストラクタ

- constructor

プロパティ

- allowDragging
- autoCollapse
- checkedItems
- childItemsPath
- controlTemplate
- displayMemberPath
- expandOnClick
- hostElement
- imageMemberPath
- isAnimated
- isContentHtml
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemsSource
- lazyLoadFunction

- nodes
- rightToLeft
- selectedItem
- selectedNode
- selectedPath
- showCheckboxes
- totalItemCount

メソッド

- addChildNode
- addEventListener
- applyTemplate
- beginUpdate
- checkAllItems
- collapseToLevel
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getControl
- getFirstNode
- getLastNode
- getNode
- getTemplate
- initialize
- invalidate
- invalidateAll
- loadTree
- onCheckedItemsChanged
- onDragEnd
- onDragOver
- onDragStart
- onDrop
- onFormatItem
- onGotFocus
- onIsCheckedChanged
- onIsCheckedChanging
- onIsCollapsedChanged
- onIsCollapsedChanging
- onItemClicked
- onItemsSourceChanged
- onLoadedItems
- onLoadingItems
- onLostFocus
- onNodeEditEnded
- onNodeEditEnding

- onNodeEditStarted
- onNodeEditStarting
- onRefreshed
- onRefreshing
- onSelectedItemChanged
- refresh
- refreshAll
- removeEventListener
- startEditing

イベント

- checkedItemsChanged
- dragEnd
- dragOver
- dragStart
- drop
- formatItem
- gotFocus
- isCheckedChanged
- isCheckedChanging
- isCollapsedChanged
- isCollapsedChanging
- itemClicked
- itemsSourceChanged
- loadedItems
- loadingItems
- lostFocus
- nodeEditEnded
- nodeEditEnding
- nodeEditStarted
- nodeEditStarting
- refreshed
- refreshing
- selectedItemChanged

コンストラクタ

constructor

```
constructor(element: any, options?): TreeView
```

TreeView クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **TreeView**

プロパティ

allowDragging

ユーザーが**TreeView** 内でノードをドラッグアンドドロップできるかどうかを指定する値を取得または設定します。

型 **boolean**

autoCollapse

ノードが展開されたときに、兄弟ノードを折りたたむかどうかを指定する値を 取得または設定します。

このプロパティは、デフォルトではtrueに設定されています。 通常は、使用していないノードを折りたたんだ方がUIがすっきりするためです。

型 **boolean**

checkedItems

現在オンに設定されている項目が含まれる配列を取得します。

返される配列には、子を持たない項目だけが含まれます。 これは、親項目のチェックボックスが子項目のオンまたはオフに 使用されるからです。

showCheckboxes プロパティと **checkedItemsChanged** プロパティも参照してください。

次に例を示します。

```
var treeViewChk = new wijmo.input.TreeView('#gsTreeViewChk', {
   displayMemberPath: 'header',
   childItemsPath: 'items',
   showCheckboxes: true,
   itemsSource: items,
   checkedItemsChanged: function (s, e) {
       var items = s.checkedItems,
           msg = '';
       if (items.length) {
           msg = '<p><b>Selected Items:</b></p><ol>\r\n';
           for (var i = 0; i < items.length; i++) {
               msg += '<li>' + items[i].header + '</li>\r\n';
           }
           msg += '</ol>';
       }
       document.getElementById('gsTreeViewChkStatus').innerHTML = msg;
   }
});
```

型 **any**[]

childItemsPath

各ノードの子項目を含む1つ以上のプロパティの名前を 取得または設定します。

このプロパティのデフォルト値は文字列'items'です。

ほとんどの場合、子項目を含むプロパティは、 ツリーのすべてのデータ項目で同じです。このような場合は、 **childItemsPath** にその名前を設定します。

ただし、項目のレベルごとに異なるプロパティを使用して 子項目を保存する場合もあります。たとえば、カテゴリ、製品、注文から成る ツリーがあるとします。その場合に、 **childItemsPath** に次のような配列を設定します。

```
// カテゴリには製品が含まれ、製品には注文が含まれます。
tree.childItemsPath = [ 'Products', 'Orders' ];
```

型 **any**

STATIC controlTemplate

TreeView コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

displayMemberPath

ノードのビジュアル表現として使用される1つ以上のプロパティの名前を 取得または設定します。

このプロパティのデフォルト値は文字列'header'です。

ほとんどの場合、ノードテキストを含むプロパティは、 ツリーのすべてのデータ項目で同じです。このような場合は、 **displayMemberPath** にその名前を設定します。

ただし、項目のレベルごとに異なるプロパティを使用して ノードテキストを表す場合もあります。たとえば、カテゴリ、製品、注文から成る ツリーがあるとします。その場合に、 **displayMemberPath** に次のような配列を設定します。

```
// カテゴリ、製品、注文のヘッダーがそれぞれ異なります。
tree.displayMemberPath = [ 'CategoryName', 'ProductName', 'OrderID' ];
```

型 **any**

expandOnClick

ユーザーがノードヘッダーをクリックしたときに、折りたたまれているノードを展開するかどうかを指定する値を取得または設定します。

型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- imageMemberPath

ノードの画像のソースとして使用する1つ以上のプロパティの名前を 取得または設定します。

型 **any**

- isAnimated

ノードを展開または折りたたむときにアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- isContentHtml

項目をプレーンテキストまたはHTMLに連結するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isReadOnly

ユーザーがノードのテキストを編集できるかどうかを指定する値を取得または設定します。

isReadOnly プロパティをfalseに設定すると、ノードに直接入力することで ツリーノードのコンテンツを編集することができます。ノードのコンテンツ全体を選択した状態で、 ［F2］キーを使用して編集モードに入ることもできます。

次のメソッドとイベントを使用して、編集動作をカスタマイズできます。

メソッド：**startEditing**、**finishEditing**

イベント：**nodeEditStarting**、**nodeEditStarted**、**nodeEditEnding**、 **nodeEditEnded**

The default value for this property is **true**.

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemsSource

TreeView 項目を含む配列を取得または設定します。

TreeView #see:itemsSource 配列は一般に、 子項目を含む項目から成る階層構造を持ちます。項目の深さに制限はありません。

たとえば、次の配列は、それぞれが2つの子ノードを持つ 3つの最上位ノードを含むツリーを生成します。

```
var tree = new wijmo.input.TreeView('#treeView', {
    displayMemberPath: 'header',
    childItemsPath: 'items',
    itemsSource: [
        { header: '1 first', items: [
            { header: '1.1 first child' },
            { header: '1.2 second child' },
        ] },
        { header: '2 second', items: [
            { header: '3.1 first child' },
            { header: '3.2 second child' },
        ] },
        { header: '3 third', items: [
            { header: '3.1 first child' },
            { header: '3.2 second child' },
        ] }
    ]
});
```

型 **any**[]

lazyLoadFunction

オンデマンドで子ノードをロードする関数を取得または設定します。

lazyLoadFunction は、2つのパラメータとして、展開されるノードと、 データが取得可能になったときに呼び出されるコールバックを 受け取ります。

コールバック関数は、**TreeView** にノードのロードプロセスが 完了したことを通知します。データのロード時にエラーがあっても、 コールバックは必ず呼び出されます。

次に例を示します。

```
var treeViewLazyLoad = new wijmo.input.TreeView('#treeViewLazyLoad', {
   displayMemberPath: 'header',
   childItemsPath: 'items',
   itemsSource: [ // 3つの遅延ロードノードから開始します
       { header: 'Lazy Node 1', items: []},
       { header: 'Lazy Node 2', items: [] },
       { header: 'Lazy Node 3', items: [] }
   ],
   lazyLoadFunction: function (node, callback) {
       setTimeout(function () { // httpの遅延をシミュレーションします
           var result = [ // 結果をシミュレーションします
               { header: 'Another lazy node...', items: [] },
               { header: 'A non-lazy node without children' },
               { header: 'A non-lazy node with child nodes', items: [
                 { header: 'hello' },
                 { header: 'world' }
               ]}
           ];
           callback(result); // コントロールに結果を返します
       }, 2500); // 2.5秒のhttpの遅延をシミュレーションします
   }
});
```

遅延ロードノードを含むツリーには、それぞれのノードにチェックボックスが ないことがある（**showCheckboxes** プロパティを参照）、**collpaseToLevel** メソッドがまだロードされていない折りたたみ ノードを展開しない、といった制限があります。

型 **Function**

nodes

現在ロードされているノードを表す**TreeNode** オブジェクトの配列を取得します。

型 **TreeNode[]**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectedItem

現在選択されているデータ項目を取得または設定します。

型 **any**

selectedNode

現在選択されている**TreeNode** を取得または設定します。

型 **TreeNode**

selectedPath

ルートから現在選択されているノードまでのすべてのノードのテキストが 入った配列を取得します。

型 **string[]**

showCheckboxes

TreeView で、ノードにチェックボックスを追加し、その状態を管理するかどうかを決定する値を 取得または設定します。

このプロパティは、遅延ロードノードがないツリーでのみ使用できます （**lazyLoadFunction** プロパティを参照）。

checkedItems プロパティと**checkedItemsChanged** イベントも参照してください。

The default value for this property is **false**.

型 **boolean**

totalItemCount

ツリー内の項目の総数を取得します。

型 **number**

メソッド

addChildNode

```
addChildNode(index: number, dataItem: any): TreeNode
```

特定の位置に子ノードを追加します。

パラメーター

- **index**: **number**
  新しい子ノードのインデックス。
- **dataItem**: **any**
  新しいノードの作成に使用されたデータ項目。

戻り値 **TreeNode**

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

checkAllItems

```
checkAllItems(check: boolean): void
```

ツリーのすべてのチェックボックスをオンまたはオフにします。

パラメーター

- **check**: **boolean**
  すべてのチェックボックスをオンまたはオフにするかどうか。

戻り値 **void**

collapseToLevel

```
collapseToLevel(level: number): void
```

すべてのツリー項目を指定されたレベルまで折りたたみます。

通常、このメソッドは一度に複数のノードを展開または折りたたみます。ただし、どのノードにも遅延ロードを実行しないので、 遅延ロードする必要がある折りたたみノードは展開されません。

パラメーター

- **level**: **number**
  表示する最大ノードレベル。

戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

戻り値 **boolean**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getFirstNode

```
getFirstNode(visible?: boolean, enabled?: boolean): TreeNode
```

TreeView の最初の**TreeNode** への参照を取得します。

パラメーター

- **visible**: **boolean** OPTIONAL
  可視のノード（祖先ノードが折りたたまれていない）だけを返すかどうか。

- **enabled**: **boolean** OPTIONAL
  有効なノード（祖先ノードが無効でない）だけを返すかどうか。

戻り値 **TreeNode**

getLastNode

```
getLastNode(visible?: boolean, enabled?: boolean): TreeNode
```

TreeView の最後の**TreeNode** への参照を取得します。

パラメーター

- **visible**: **boolean** OPTIONAL
  可視のノード（祖先ノードが折りたたまれていない）だけを返すかどうか。

- **enabled**: **boolean** OPTIONAL
  有効なノード（祖先ノードが無効でない）だけを返すかどうか。

戻り値 **TreeNode**

getNode

```
getNode(item: any): TreeNode
```

特定のデータ項目を表す**TreeNode** オブジェクトを取得します。

パラメーター

- **item**: **any**
  検索するデータ項目。

戻り値 **TreeNode**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

loadTree

```
loadTree(preserveOutlineState?: boolean): void
```

現在の**itemsSource** のデータを使用してツリーをロードします。

パラメーター

- **preserveOutlineState**: **boolean** OPTIONAL
  ツリーのデータをロードするときにアウトラインの状態を保持するかどうか。 デフォルト値はfalseです。

戻り値 **void**

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

checkedItemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onDragEnd

```
onDragEnd(e?: EventArgs): void
```

dragEnd イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onDragOver

```
onDragOver(e: TreeNodeDragDropEventArgs): boolean
```

dragOver イベントを発生させます。

パラメーター

- **e**: **TreeNodeDragDropEventArgs**
  イベントデータを含む**TreeNodeDragDropEventArgs**。

戻り値 **boolean**

onDragStart

```
onDragStart(e: TreeNodeEventArgs): boolean
```

dragStart イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

戻り値 **boolean**

onDrop

```
onDrop(e: TreeNodeDragDropEventArgs): boolean
```

drop イベントを発生させます。

パラメーター

- **e**: **TreeNodeDragDropEventArgs**
  イベントデータを含む**TreeNodeDragDropEventArgs**。

戻り値 **boolean**

onFormatItem

```
onFormatItem(e: FormatNodeEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatNodeEventArgs**
  イベントデータを含む**FormatNodeEventArgs**。

戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsCheckedChanged

```
onIsCheckedChanged(e: TreeNodeEventArgs): void
```

isCheckedChanged イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

戻り値 **void**

onIsCheckedChanging

```
onIsCheckedChanging(e: TreeNodeEventArgs): boolean
```

isCheckedChanging イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

戻り値 **boolean**

onIsCollapsedChanged

```
onIsCollapsedChanged(e: TreeNodeEventArgs): void
```

isCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

戻り値 **void**

onIsCollapsedChanging

```
onIsCollapsedChanging(e: TreeNodeEventArgs): boolean
```

isCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

戻り値 **boolean**

onItemClicked

```
onItemClicked(e?: EventArgs): void
```

itemClicked イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLoadedItems

```
onLoadedItems(e?: EventArgs): void
```

loadedItems イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLoadingItems

```
onLoadingItems(e?: CancelEventArgs): boolean
```

loadingItems イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs** OPTIONAL

戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onNodeEditEnded

```
onNodeEditEnded(e: TreeNodeEventArgs): void
```

nodeEditEnded イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

戻り値 **void**

onNodeEditEnding

```
onNodeEditEnding(e: TreeNodeEventArgs): boolean
```

nodeEditEnding イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

戻り値 **boolean**

onNodeEditStarted

```
onNodeEditStarted(e: TreeNodeEventArgs): void
```

nodeEditStarted イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

戻り値 **void**

onNodeEditStarting

```
onNodeEditStarting(e: TreeNodeEventArgs): boolean
```

nodeEditStarting イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

戻り値 **boolean**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedItemChanged

```
onSelectedItemChanged(e?: EventArgs): void
```

selectedItemChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

ツリーを再作成するためにオーバーライドされます。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示します。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **`Control`**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**`Control`** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **`Control`**
戻り値 **number**

startEditing

```
startEditing(node?: TreeNode): boolean
```

特定の**TreeNode** の編集を開始します。

パラメーター

- **node**: **TreeNode** OPTIONAL
  編集する**TreeNode**。 指定しない場合は、現在選択されているノードが使用されます。

戻り値 **boolean**

イベント

checkedItemsChanged

checkedItems プロパティの値が変化すると発生します。

引数 **EventArgs**

dragEnd

マウスまたはキーボードでノードを別の位置にドロップするか、操作をキャンセルして、 ドラッグ/ドロップ操作を終了すると発生します。

引数 **EventArgs**

dragOver

ユーザーが**TreeView** で1つのノードを他のノードの上にドラッグしているときに発生します。

このイベントは、**allowDragging** プロパティがtrueに設定されている場合にのみ発生します。

イベントの**cancel**パラメータをtrueに設定することで、 特定のノードや位置の上にドロップできないようにすることができます。

引数 **TreeNodeDragDropEventArgs**

dragStart

ユーザーが行のドラッグを開始するときに発生します。

このイベントは、**allowDragging** プロパティがtrueに設定されている場合にのみ発生します。

イベントの**cancel**パラメータをtrueに設定することで、 ノードをドラッグできないようにすることができます。

引数 **TreeNodeEventArgs**

drop

ユーザーが**TreeView** でノードをドロップしたときに発生します。

引数 **TreeNodeDragDropEventArgs**

formatItem

ノードを表す要素が作成されたときに発生します。

このイベントを使用して、表示するノードを書式設定できます。

次の例は、**formatItem**イベントを使用して、ツリーの新しい項目の右側に "new"バッジを追加します。

```
var treeViewFmtItem = new wijmo.input.TreeView('#treeViewFmtItem', {
    displayMemberPath: 'header',
    childItemsPath: 'items',
    itemsSource: items,
    formatItem: function (s, e) {
        if (e.dataItem.newItem) {
            e.element.innerHTML +=
                '<img style="margin-left:6px" src="resources/new.png"/>';
        }
    }
});
```

引数 **FormatNodeEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isCheckedChanged

isChecked プロパティの値が変化した後に発生します。

引数 **TreeNodeEventArgs**

isCheckedChanging

isChecked プロパティの値が変化する前に発生します。

引数 **TreeNodeEventArgs**

isCollapsedChanged

isCollapsed プロパティの値が変化した後に発生します。

引数 **TreeNodeEventArgs**

isCollapsedChanging

isCollapsed プロパティの値が変化する前に発生します。

引数 **TreeNodeEventArgs**

itemClicked

ユーザーが項目をクリックするか、［Enter］キーを押して、項目が選択されたときに発生します。

通常、このイベントはナビゲーションツリーで使用されます。 **selectedItem** プロパティを使用して、 クリックされた項目を取得します。

引数 **EventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

引数 **EventArgs**

loadedItems

ツリー項目が生成された後に発生します。

引数 **EventArgs**

loadingItems

ツリー項目が生成される前に発生します。

引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

nodeEditEnded

TreeNode が編集モードを出た後に発生します。

引数 **TreeNodeEventArgs**

nodeEditEnding

TreeNode が編集モードを出る前に発生します。

引数 **TreeNodeEventArgs**

nodeEditStarted

TreeNode が編集モードに入った後に発生します。

引数 **TreeNodeEventArgs**

nodeEditStarting

TreeNode が編集モードに入る前に発生します。

引数 **TreeNodeEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedItemChanged

selectedItem プロパティの値が変化すると発生します。

引数 **EventArgs**

DropPosition 列挙体

ファイル wijmo.nav.js
モジュール **wijmo.nav**

ドラッグアンドドロップ操作中に**TreeNode** をドロップする位置を指定します。

メンバー

名前	値	説明
Before	0	このノードは、ターゲットノードの前の兄弟になります。
After	1	このノードは、ターゲットノードの次の兄弟になります。
Into	2	このノードは、ターゲットノードの最後の子になります。

wijmo.grid.sheet モジュール

ファイル `wijmo.grid.sheet.js`
モジュール **wijmo.grid.sheet**

FlexSheet コントロールとその関連クラスを定義します。

FlexSheet コントロールは、**FlexGrid** コントロールを拡張して、 Excelと同様の機能を提供します。

サンプル

Show me (https://jsfiddle.net/Wijmo5/t8tp9tnx)

クラス

- ColumnSortDescription
- DefinedName
- DraggingRowColumnEventArgs
- FlexSheet
- FlexSheetColumnFilter
- FlexSheetColumnFilterEditor
- FlexSheetConditionFilter
- FlexSheetFilter
- FlexSheetPanel
- FlexSheetValueFilter
- FlexSheetValueFilterEditor
- HeaderRow
- RowColumnChangedEventArgs
- Sheet
- SheetCollection
- SortManager
- Table
- TableColumn
- TableStyle
- UndoStack
- UnknownFunctionEventArgs

インターフェイス

- IBandedTableSectionStyle
- ICellStyle
- IColumnFilterSetting
- IConditionFilterSetting
- IFilterSetting
- IFlexSheetXlsxOptions
- IFormatState
- ITableOptions
- ITableSectionStyle
- IValueFiterSetting

列挙体

- TableSection

ColumnSortDescription クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

FlexSheet の列のソート基準を記述します。

コンストラクタ

- constructor

プロパティ

- ascending
- columnIndex

メソッド

- clone

コンストラクタ

constructor

```
constructor(columnIndex: number, ascending: boolean): ColumnSortDescription
```

ColumnSortDescription クラスの新しいインスタンスを初期化します。

パラメーター

- **columnIndex**: **number**
  ソート基準となる列を示します。
- **ascending**: **boolean**
  ソート順序。

戻り値 **ColumnSortDescription**

プロパティ

- ascending

ソート順序を昇順にするかどうかを取得または設定します。

型 **boolean**

- columnIndex

列インデックスを取得または設定します。

型 **number**

メソッド

clone

```
clone(): ColumnSortDescription
```

ColumnSortDescriptionのコピーを作成します。

戻り値　　**ColumnSortDescription**

DefinedName クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

FlexSheetの定義済みの名前項目を表します。

コンストラクタ

- constructor

プロパティ

- name
- sheetName
- value

コンストラクタ

constructor

```
constructor(owner: FlexSheet, name: string, value: any, sheetName?: string): DefinedName
```

DefinedNameクラスの新しいインスタンスを初期化します。

パラメーター

- **owner**: **FlexSheet**
  オーナーである**FlexSheet** コントロール。
- **name**: **string**
  定義された名前項目の名前。
- **value**: **any**
  定義された名前項目の値。
- **sheetName**: **string** OPTIONAL
  シート名は、定義された名前項目がFlexSheetのどのシートに機能するかを示します。 省略した場合、定義された名前の項目はFlexSheetのすべてのシートで機能します。

戻り値 **DefinedName**

プロパティ

- name

定義された名前項目の名前を取得または設定します。

型 **string**

- sheetName

定義された名前項目のsheetNameを取得します。

型 **string**

value

定義された名前項目の値を取得または設定します。

型 **any**

DraggingRowColumnEventArgs クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

draggingRowColumn イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- empty
- isDraggingRows
- isShiftKey

コンストラクタ

constructor

```
constructor(isDraggingRows: boolean, isShiftKey: boolean): DraggingRowColumnEventArgs
```

DraggingRowColumnEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **isDraggingRows**: **boolean**
  Draggingイベントが行と列のどちらのドラッグによって発生したかを示します。
- **isShiftKey**: **boolean**
  ドラッグ中に［Shift］キーが押されているかどうかを示します。

戻り値 **DraggingRowColumnEventArgs**

プロパティ

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

- isDraggingRows

このイベントが行のドラッグと列のドラッグのどちらを表しているかを示す値を取得します。

型 **boolean**

- isShiftKey

［Shift］キーが押されているかどうかを示す値を取得します。

型 **boolean**

FlexSheet クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **FlexGrid**
派生クラス **WjFlexSheet**
表示 ☐継承されたメンバー ☐イベント発生元

FlexSheet コントロールを定義します。

FlexSheet コントロールは、**FlexGrid** コントロールを拡張して、計算エンジン、複数シート、元に戻す/やり直し、XLSXインポート/エクスポートなど、Excelと同様の機能を提供します。

features such as a calculation engine, multiple sheets, undo/redo, and XLSX import/export.

A complete list of the functions supported by the **FlexSheet**'s calculation engine can be found here: FlexSheet Functions.

サンプル

Show me (https://jsfiddle.net/Wijmo5/t8tp9tnx)

コンストラクタ

- constructor

プロパティ

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoScroll
- autoSearch
- autoSizeMode
- bottomLeftCells
- cellFactory
- cells
- childItemsPath
- clientSize
- cloneFrozenCells
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- deferResizing
- definedNames
- editableCollectionView

- editRange
- enableDragDrop
- enableFormulas
- filter
- frozenColumns
- frozenRows
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- isDisabled
- isFunctionListOpen
- isReadOnly
- isTabHolderVisible
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemValidator
- keyActionEnter
- keyActionTab
- mergeManager
- newRowAtTop
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- rightToLeft
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollSize
- selectedItems
- selectedRows
- selectedSheet
- selectedSheetIndex
- selection
- selectionMode
- sheets
- showAlternatingRows
- showDropDown
- showErrors
- showFilterIcons
- showGroups
- showMarquee
- showSelectedHeaders
- showSort
- sortManager
- sortRowIndex

- stickyHeaders
- topLeftCells
- treeIndent
- undoStack
- validateEdits
- viewRange
- virtualizationThreshold

メソッド

- addBoundSheet
- addEventListener
- addFunction
- addUnboundSheet
- applyCellsStyle
- applyFunctionToCell
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- clear
- collapseGroupsToLevel
- containsFocus
- convertNumberToAlpha
- deferUpdate
- deleteColumns
- deleteRows
- dispose
- disposeAll
- endUpdate
- evaluate
- finishEditing
- focus
- freezeAtCursor
- getBuiltInTableStyle
- getCellBoundingRect
- getCellData
- getCellValue
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getSelectionFormatState
- getTemplate
- hideFunctionList

- hitTest
- initialize
- insertColumns
- insertRows
- invalidate
- invalidateAll
- isRangeValid
- load
- loadAsync
- mergeRange
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onColumnChanged
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowColumn
- onDraggingRowOver
- onDroppingRowColumn
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onItemsSourceChanging
- onLoaded
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onPrepareChangingColumn
- onPrepareChangingRow
- onRefreshed
- onRefreshing

- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowChanged
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectedSheetChanged
- onSelectionChanged
- onSelectionChanging
- onSheetCleared
- onSortedColumn
- onSortingColumn
- onUnknownFunction
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- redo
- refresh
- refreshAll
- refreshCells
- removeEventListener
- save
- saveAsync
- scrollIntoView
- select
- selectNextFunction
- selectPreviousFunction
- setCellData
- setClipString
- showColumnFilter
- showFunctionList
- startEditing
- toggleDropDownList
- undo

イベント

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding

- columnChanged
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowColumn
- draggingRowOver
- droppingRowColumn
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- itemsSourceChanging
- loaded
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- prepareChangingColumn
- prepareChangingRow
- refreshed
- refreshing
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowChanged
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectedSheetChanged
- selectionChanged
- selectionChanging
- sheetCleared
- sortedColumn
- sortingColumn

- unknownFunction
- updatedLayout
- updatedView
- updatingLayout
- updatingView

コンストラクタ

constructor

```
constructor(element: any, options?): FlexSheet
```

FlexSheet クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **FlexSheet**

プロパティ

activeEditor

現在アクティブになっているセルエディタを表す**HTMLInputElement**を取得します。

継承元 **FlexGrid**
型 **HTMLInputElement**

allowAddNew

ユーザーがソースコレクションに項目を追加できるようにグリッドに新規行テンプレートを表示するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、新規行テンプレートは表示されません。

継承元 **FlexGrid**
型 **boolean**

allowDelete

ユーザーが［Delete］キーを押したときにグリッドの選択されている行を削除するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、選択されている行は削除されません。

継承元 **FlexGrid**
型 **boolean**

allowDragging

ユーザーがマウスを使用して行、列、またはその両方をドラッグできるかどうかを決定する値を取得または設定します。

autoScroll プロパティがtrueに設定されている場合、ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされます。

グリッドでは、列のドラッグがデフォルトで有効になっています。

グリッドが連結モードでは、行をドラッグするには特別な 考慮が必要になります。

グリッドが連結モードでは、行をドラッグするとデータソースとの同期が失われます（たとえば行4は6行として参照されることがあります）。 これを回避するには、**draggedRow** イベントを処理し、データを新しい行の位置と同期させる必要があります。

また、**allowSorting** プロパティをfalseに設定する必要があります。そうしないと、行の順序がデータによって決定され、行をドラッグするのは無意味になります。

次のフィドルでは、連結グリッドでの行のドラッグを実行しています。 **Bound Row Dragging (http://jsfiddle.net/Wijmo5/kyg0qsda/)**.

継承元 **FlexGrid**
型 **AllowDragging**

allowMerging

グリッドのどの部分のセル結合を許可するかを取得または設定します。

継承元 **FlexGrid**
型 **AllowMerging**

allowResizing

ユーザーがマウスを使用して行、列、またはその両方をサイズ変更できるかどうかを決定する値を取得または設定します。

サイズ変更が可能な場合は、列ヘッダーセルの右端をドラッグして列を、行ヘッダーセルの下端をドラッグして行をサイズ変更できます。

ヘッダーセルの端をダブルクリックして、行および列をコンテンツに合わせて自動的にサイズ変更することもできます。 自動サイズ変更の動作は、**autoSizeMode** プロパティを使用してカスタマイズできます。

継承元 **FlexGrid**
型 **AllowResizing**

allowSorting

ユーザーが列ヘッダーセルをクリックして列をソートできるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

autoClipboard

グリッドがクリップボードショートカットを処理するかどうかを決定する値を取得または設定します。

次のクリップボードショートカットがあります。

［Ctrl］+［C］、［Ctrl］+［Ins］	グリッドの選択範囲をクリップボードにコピーします。
［Ctrl］+［V］、［Shift］+［Ins］	クリップボードのテキストをグリッドの選択範囲に貼り付けます。

クリップボード操作は、表示されている行および列だけが対象となります。

読み取り専用のセルは、貼り付け操作の影響を受けません。

継承元 **FlexGrid**
型 **boolean**

autoGenerateColumns

グリッドが**itemsSource** に基づいて列を自動的に生成するかどうかを 決定する値を取得または設定します。

列の生成は、少なくとも1項目を含む**itemsSource** プロパティに依存します。このデータ項目が検査され、列が作成されて、プリミティブ値（数値、文字列、Boolean、またはDate）を含むプロパティに連結されます。

プロパティをnullに設定すると、適切なタイプを推測する方法がないため、列は生成されません。このような場合は、**autoGenerateColumns** プロパティをfalseに設定し、明示的に列を作成する必要があります。次に例を示します。

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // データ項目にnull値が含まれる場合があります
  columns: [                  // そのため、列を明示的に定義します
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

継承元 **FlexGrid**
型 **boolean**

autoScroll

ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされるかどうかを決定する値を取得または設定します。

行と列のドラッグは、**allowDragging** プロパティによって制御されます。

このプロパティはデフォルトでtrueに設定されます。

継承元 **FlexGrid**
型 **boolean**

autoSearch

ユーザーが読み取り専用セルに入力するに伴ってグリッドでセルを検索するかどうかを決定する値を取得または設定します。

検索が現在選択されている列(編集不可能な場合)で行われます。検索は現在選択されている行から始まり、大文字と小文字を区別されません。

継承元 **FlexGrid**
型 **boolean**

autoSizeMode

行または列のサイズを自動設定するときにどのセルを考慮に入れるかを取得または設定します。

このプロパティは、ユーザーが列ヘッダの端をダブルクリックしたときの動作を制御します。

デフォルトでは、その列のヘッダセルとデータセルの内容に基づいて列の幅が自動的に設定されます。このプロパティを使用すると、ヘッダのみまたはデータのみに基づいて列の幅を設定するように変更できます。

継承元 **FlexGrid**
型 **AutoSizeMode**

bottomLeftCells

左下のセルを含む**GridPanel** を取得します。

bottomLeftCells パネルは、行ヘッダの下、**columnFooters** パネルの左に表示されます。

継承元 **FlexGrid**
型 **GridPanel**

cellFactory

このグリッドのセルを作成および更新する**CellFactory** を取得または設定します。

継承元 **FlexGrid**
型 **CellFactory**

cells

データセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

childItemsPath

階層グリッドで子の行の生成に使用される 1 つ以上のプロパティの名前を取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `childItemsPath = 'items'` ）。

項目の異なるレベルに異なる名前を持つ子項目がある場合は、このプロパティに、各レベルの子項目を含むプロパティの名前から成る配列を設定します。（たとえば、 `childItemsPath = ['checks','earnings'];` ）。

サンプル

Show me (https://jsfiddle.net/Wijmo5/kk9j93bL)

継承元 **FlexGrid**
型 **any**

clientSize

コントロールのクライアントサイズを取得します（コントロールのサイズからヘッダーとスクロールバーを引いた値）。

継承元 **FlexGrid**
型 **Size**

cloneFrozenCells

スクロール中に知覚されるパフォーマンスを向上させるには、FlexGridがフリーズされたセルを複製し、別の要素に表示するかどうかを決定する値を取得または設定します。

このプロパティのデフォルト値はnullであり、ブラウザーによって一番適当な設定が選択されます。

継承元 **FlexGrid**
型 **boolean**

collectionView

グリッドデータを含む**ICollectionView** を取得します。

継承元 **FlexGrid**
型 **ICollectionView**

columnFooters

列フッターセルを保持する**GridPanel** を取得します。

columnFooters パネルは、グリッドセルの下、**bottomLeftCells** パネルの右に表示されます。これを使用して、グリッドデータの下にサマリー情報を表示できます。

以下の例では、**columnFooters** パネルに 1行を追加して、**aggregate** プロパティが設定された列に サマリーデータを表示する方法を示します。

```
function addFooterRow(flex) {

  // 集計を表示するGroupRowを作成します
  var row = new wijmo.grid.GroupRow();

  // 列フッターパネルに行を追加します
  flex.columnFooters.rows.push(row);

  // ヘッダーにシグマを表示します
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

継承元 **FlexGrid**
型 **GridPanel**

columnHeaders

列ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

columnLayout

現在の列レイアウトを定義するJSON文字列を取得または設定します。

列レイアウト文字列は、列とそのプロパティを含む配列を表します。これを使用して、ユーザーが定義した列レイアウトをセッションを越えて保持できます。また、ユーザーが列レイアウトを変更できるアプリケーションで元に戻す/やり直し機能を実装することもできます。

データマップはシリアル化できないため、列レイアウト文字列に**dataMap**　プロパティは含まれていません。

継承元 **FlexGrid**
型 **string**

columns

グリッドの列コレクションを取得します。

継承元 **FlexGrid**
型 **ColumnCollection**

controlRect

コントロールの外接矩形（ページ座標単位）を取得します。

継承元 **FlexGrid**
型 **Rect**

STATIC controlTemplate

Overrides the template used to instantiate **FlexSheet** control.

型 **any**

deferResizing

ユーザーがマウスボタンを放すまで、行および列のサイズ変更を遅らせるかどうかを決定する値を取得または設定します。

デフォルトでは、**deferResizing** はfalseに設定されており、ユーザーがマウスをドラッグするに伴って行および列がサイズ変更されます。このプロパティをtrueに設定すると、グリッドにサイズ変更マーカーが表示され、ユーザーがマウスボタンを放したときに初めて行または列のサイズが変更されます。

継承元 **FlexGrid**
型 **boolean**

definedNames

FlexSheetで定義された名前付き範囲または名前付き式を表す**IDefinedName** オブジェクトの配列を取得します。

型 **ObservableArray**

editableCollectionView

グリッドデータを含む**IEditableCollectionView** を取得します。

継承元 **FlexGrid**
型 **IEditableCollectionView**

editRange

現在編集中のセルを識別する**CellRange** を取得します。

継承元 **FlexGrid**
型 **CellRange**

enableDragDrop

Gets or sets the value to indicates whether enable drag and drop rows or columns in FlexSheet.

型 **boolean**

enableFormulas

Gets or sets the value to indicates whether enable formulas in FlexSheet.

型 **boolean**

filter

FlexSheetのフィルタリングを制御する**FlexSheetFilter** インスタンスを取得します。

型 **FlexSheetFilter**

frozenColumns

固定された列の数を取得または設定します。

固定された列は水平方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

frozenRows

静止行の数を取得または設定します。

固定された行は垂直方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

groupHeaderFormat

グループヘッダーコンテンツを作成するために使用される書式文字列を取得または設定します。

この文字列は、任意のテキストと次の置換文字列を含むことができます。

- **{name}**：グループ化されるプロパティの名前。
- **{value}**：グループ化されるプロパティの値。
- **{level}**：グループレベル。
- **{count}**：このグループ内にある項目の合計数。

列がグループ化プロパティに連結されている場合は、列ヘッダーを使用して パラメータを置換し、列の書式とデータマップを使用して `{value}` パラメータを計算します。 列がない場合は、代わりにグループ情報が使用されます。

グループプロパティに連結された非表示の列を追加して、グループヘッダーセルの書式設定をカスタマイズすることもできます。

このプロパティのデフォルト値は
`'{name}: <b>{value}</b>({count:n0} items)'` です。 これは、 `'Country: UK (12 items)'` や
`'Country: Japan (8 items)'` のようなグループヘッダーを作成します。

継承元 **FlexGrid**
型 **string**

headersVisibility

行ヘッダおよび列ヘッダが表示されるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- imeEnabled

編集モードでないときに、グリッドがIME（Input Method Editor）をサポートするかどうかを決定する値を取得または設定します。

このプロパティは、日本語、中国語、韓国語など、IMEサポートを必要とする言語のサイト/アプリケーションにのみ関係します。

継承元 **FlexGrid**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isFunctionListOpen

関数リストが開いているかどうかを示す値を取得します。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してセル値を変更できるかどうかを判定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

- isTabHolderVisible

TabHolderが表示されているかどうかを示す値を取得または設定します。

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

このグリッドのセルのカスタマイズに使用されるフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意のセルに任意の内容を追加できます。そのため、グリッドセルの外観と動作を非常に柔軟に制御することが可能です。

この関数は、セルを含む**GridPanel**、セルの行インデックスと列インデックス、セルを表すHTML要素の4つのパラメーターをとります。通常は、関数内でセル要素の**innerHTML**　プロパティを変更するようにします。

例:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == wijmo.grid.CellType.Cell) {

    // セルにスパークラインを描画します。
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

FlexGridはセルをリサイクルするため、**itemFormatter** によってセルのスタイル属性を変更する場合は、 セルの適切でないスタイル属性を事前にリセットする必要があります。例:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // カスタマイズする属性をリセットします。
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // このセルのcolorおよびbackgroundColor属性をカスタマイズします。
  ...
}
```

複数のクライアントがグリッドのレンダリングをカスタマイズできるようにする場合は（たとえば、ディレクティブや再利用可能なライブラリを作成するときなど）、代わりに**formatItem** イベントを使用することを検討してください。このイベントを使用すると、複数のクライアントがそれぞれ独自のハンドラをアタッチできます。

継承元　　**FlexGrid**
型　　**Function**

itemsSource

グリッドに表示される項目を含む配列または**ICollectionView** を取得または設定します。

継承元　　**FlexGrid**
型　　**any**

itemValidator

セルに有効なデータが含まれているかどうかを決定するバリデータ関数を取得または設定します。

指定された場合、バリデータ関数はセルの行インデックスと列インデックスを含む2つのパラメータをとり、エラーの説明を含む文字列を返す必要があります。

このプロパティは、バインドされていないグリッドを処理する場合に特に便利です。バインドされたグリッドは、代わりに**getError** プロパティを使用して検証できます。

この例では、セルのすぐ上のセルと同じデータがセルに含まれないようにする方法を示します。

```
// 上のセルは、現在のセルと同じ値が含まれていないことを確認します
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'これは重複した値です...'
    }
  }
  return null; // エラーなし
}
```

継承元 **FlexGrid**
型 **Function**

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は**MoveDown** となり、コントロールがカーソルを次の行に移動します。 この動作は標準的なExcel式の動作です。

継承元 **FlexGrid**
型 **KeyAction**

keyActionTab

［Tab］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は、**None** となります。これにより、Tabキーが押されたときにブラウザ上で次または前のコントロールを選択できます。これは、ページのアクセシビリティを向上させるために推奨される設定になります。

以前のバージョンでは、デフォルトは**Cycle** に設定されていました。これにより、コントロールがカーソルを、グリッドを横切って下部に移動しました。これは標準的なExcelの動作ですが、アクセシビリティには適していません。

また、**CycleOut** の設定が存在します。これにより、カーソルがセルを通じて移動（**Cycle** ）してから、最後または最初のセルが選択されたときにページの次/前のコントロールに移動します 。

継承元 **FlexGrid**
型 **KeyAction**

mergeManager

セルの結合方法を決定する**MergeManager** オブジェクトを取得または設定します。

継承元 **FlexGrid**
型 **MergeManager**

newRowAtTop

新しい行テンプレートをグリッドの上部に配置するか、下部に配置するかを示す値を取得または設定します。

newRowAtTop プロパティをtrueに設定した場合、新しい行テンプレートを常に表示したままにする場合は、**frozenRows** プロパティを 1に設定します。これにより、新しい行テンプレートは上部に固定され、ビューからスクロールオフされなくなります。

allowAddNew プロパティがtrueに設定されている場合、および**itemsSource** オブジェクトが新しい項目の追加をサポートしている場合にのみ、新しい行テンプレートが表示されます。

継承元 **FlexGrid**
型 **boolean**

preserveOutlineState

データがリフレッシュされたときに、グリッドがノードの展開/折りたたみ状態を保持するかどうかを決定する値を取得または設定します。

preserveOutlineState プロパティの実装は、JavaScriptの**Map** オブジェクトに基づいています。このオブジェクトはIE 9およびIE 10で利用できません。

継承元 **FlexGrid**
型 **boolean**

preserveSelectedState

データがリフレッシュされたときに、グリッドが行の選択状態を保持するかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

quickAutoSize

グリッドが列のサイズを自動調整するときに精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、自動サイズ変更の機能が無効になります。このプロパティをtrueに設定すると、各列の**quickAutoSize** プロパティの値に従って、その機能が有効になります。null（デフォルト値）に設定した場合、カスタムの**itemFormatter** を持たないグリッドの機能、または**itemFormatter** イベントにアタッチされたハンドラの機能が有効になります。

継承元 **FlexGrid**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rowHeaderPath

行ヘッダーセルを作成するために使用されるプロパティの名前を取得または設定します。

行ヘッダーセルは表示されないし、選択することもできません。アクセシビリティツールと組み合わせて使用することを意図しています。

継承元 **FlexGrid**
型 **string**

rowHeaders

行ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

rows

グリッドの行コレクションを取得します。

継承元 **FlexGrid**
型 **RowCollection**

scrollPosition

グリッドのスクロールバーの値を表す**Point** を取得または設定します。

継承元 **FlexGrid**
型 **Point**

scrollSize

グリッド内容のサイズ（ピクセル単位）を取得します。

継承元 **FlexGrid**
型 **Size**

selectedItems

現在選択されているデータ項目を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selectedRows

現在選択されている行を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selectedSheet

FlexSheetで現在選択されている**Sheet** を取得します。

型 **Sheet**

- selectedSheetIndex

FlexSheet で現在選択されているシートのインデックスを取得または設定します。

型 **number**

- selection

現在の選択を取得または設定します。

継承元 **FlexGrid**
型 **CellRange**

- selectionMode

現在の選択モードを取得または設定します。

継承元 **FlexGrid**
型 **SelectionMode**

- sheets

ワークブックのシートを表す**Sheet** オブジェクトのコレクションを取得します。

型 **SheetCollection**

- showAlternatingRows

グリッドで交互表示行のセルに'wj-alt'クラスを追加するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、CSSを変更しなくても、交互表示行スタイルを無効にできます。

継承元 **FlexGrid**
型 **boolean**

- showDropDown

グリッドで、**showDropDown** プロパティがtrueに設定されている列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **FlexGrid**
型 **boolean**

- showErrors

グリッドで、検証エラーがあるセルとエラーの説明を含むツールチップに'wj-state-invalid' クラスを追加するかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**itemValidator** または**getError** プロパティを使用して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

showFilterIcons

フィルタアイコンの表示/非表示を取得または設定します。

型 **boolean**

showGroups

データグループを区切るためにグリッドにグループ行を挿入するかどうかを決定する値を 取得または設定します。

データグループを作成するには、グリッドの**itemsSource** として使用される**ICollectionView** オブジェクトの**groupDescriptions** プロパティを変更します。

継承元 **FlexGrid**
型 **boolean**

showMarquee

現在の選択範囲の周囲にマーキー要素を表示するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

showSelectedHeaders

選択されているヘッダセルを示すためにクラス名を追加するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

showSort

グリッドで、列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

ソートは、グリッドの**itemsSource** として使用される**ICollectionView** オブジェクトの**sortDescriptions** プロパティによって制御されます。

継承元 **FlexGrid**
型 **boolean**

sortManager

FlexSheetのソートを制御する**SortManager** インスタンスを取得します。

型 **SortManager**

sortRowIndex

ソートインジケーターを表示する列ヘッダパネルの行のインデックスを取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、 **columnHeaders** パネルの最後の行がソート行になります。

継承元 **FlexGrid**
型 **number**

stickyHeaders

ユーザーがドキュメントをスクロールしたときに、列ヘッダーを表示したままにするかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

topLeftCells

左上のセル（列ヘッダーの左）を含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

treeIndent

異なるレベルの行グループをオフセットするインデントを取得または設定します。

継承元 **FlexGrid**
型 **number**

undoStack

FlexSheet の元に戻す/やり直し操作を制御する**UndoStack** インスタンスを取得します。

型 **UndoStack**

validateEdits

検証に失敗した編集をユーザーがコミットしようとしたときに、グリッドを編集モードのままにするかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**getError** メソッドを呼び出して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

viewRange

現在表示されているセルの範囲を取得します。

継承元 **FlexGrid**
型 **CellRange**

virtualizationThreshold

仮想化を有効にするために必要な最小行数、最小列数、またはその両方を取得または設定します。

このプロパティはデフォルトでゼロに設定されます。つまり、仮想化は常に有効ということです。これにより、バインディングパフォーマンスとメモリー必要量が向上しますが、スクロール時のパフォーマンス低下は犠牲になります。

グリッドの行数が少ない場合（約50〜100）、このプロパティを150などのやや高い値に設定することで、スクロールのパフォーマンスを向上させることができます。これにより、仮想化が無効になり連結処理が遅くなりますが、認識されるスクロールのパフォーマンスが向上する可能性があります。たとえば、以下のコードは、データソースに150を超える項目がある場合、グリッドがセルを仮想化します。

```
// 150を超える項目がある場合はグリッドを仮想化します
theGrid.virtualizationThreshold = 150;
```

このプロパティを200より大きい値に設定することは推奨されません。ロード処理の時間が長くなり、グリッドはすべての行のセルを作成しているときに数秒間フリーズするため、ページ上の要素数が多いためブラウザが遅くなります。

行と列に別々の仮想化しきい値を設定する場合は、**virtualizationThreshold** プロパティを2つの数値を含む配列に設定できます。この場合、最初の数値は行のしきい値として使用され、2番目の数値は列のしきい値として使用されます。たとえば、次のコードでは、グリッドは行を仮想化しますが、列を仮想化しません。

```
// 行（しきい値0）は仮想化しますが、列は仮想化しません（しきい値10,000）
theGrid.virtualizationThreshold = [0, 10000];
```

継承元 **FlexGrid**
型 **any**

メソッド

addBoundSheet

```
addBoundSheet(sheetName: string, source: any, pos?: number, grid?: FlexGrid): Sheet
```

バインドされた**Sheet** を**FlexSheet**に追加します。

パラメーター

- **sheetName**: **string**
  Sheet の名前。
- **source**: **any**
  Sheet の項目ソース。
- **pos**: **number** OPTIONAL
  sheetsコレクション内の位置。
- **grid**: **FlexGrid** OPTIONAL
  Sheet に関連付ける**FlexGrid** インスタンス。 これを指定しない場合、新しい**FlexGrid** インスタンスが作成されます。

戻り値 **Sheet**

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

addFunction

```
addFunction(name: string, func: Function, description?: string, minParamsCount?: number, maxParamsCount?: number): void
```

FlexSheet にカスタム関数を追加します。

パラメーター

- **name**: **string**
  カスタム関数の名前。

- **func**: **Function**
  カスタム関数。
  関数のシグネチャは次のようになります。

  ```
  function (...params: any[][][]): any;
  ```

  関数は可変数のパラメータをとり、各パラメータは関数の引数として渡される式に対応します。 関数の引数として渡される式が単一の値またはセル範囲に解決されるかどうかにかかわらず、各パラメーター値は常に解決された値の2次元配列となります。 配列内の行数（第1のインデックス）および列数（第2のインデックス）は、指定されたセル範囲のサイズに対応します。 引数が単一の値に解決される式である場合、その値はparam[0][0]インデクサを使用して取り出すことができる1対1の配列になります。
  以下のコードでは、カスタムのSumproduct関数('customSumProduct')をFlexSheetに追加しています。

  ```
  flexSheet.addFunction('customSumProduct', (...params: any[][][]) => {
     let result = 0,
         range1 = params[0],
         range2 = params[1];

     if (range1.length > 0 && range1.length === range2.length && range1[0].length === range2[0].length) {
         for (let i = 0; i < range1.length; i++) {
             for (let j = 0; j < range1[0].length; j++) {
                 result += range1[i][j] * range2[i][j];
             }
         }
     }
     return result;
  }, 'Custom SumProduct Function', 2, 2);
  ```

  上記の関数を追加した後は、以下のようなシートセル式で使用できます。

  ```
  =customSumProduct(A1:B5, B1:C5)
  ```

- **description**: **string** OPTIONAL
  カスタム関数の説明。**FlexSheet** の自動関数補完に表示されます。

- **minParamsCount**: **number** OPTIONAL
  関数が必要とするパラメータの最小数。

- **maxParamsCount**: **number** OPTIONAL
  関数が必要とするパラメータの最大数。カスタム関数のパラメータ数が任意である場合は、minParamsCountおよびmaxParamsCountをnullに設定する必要があります。

戻り値 **void**

addUnboundSheet

```
addUnboundSheet(sheetName?: string, rows?: number, cols?: number, pos?: number, grid?: FlexGrid): Sheet
```

バインドされていない**Sheet** を**FlexSheet**に追加します。

パラメーター

- **sheetName**: **string** OPTIONAL
  Sheetの名前。
- **rows**: **number** OPTIONAL
  Sheetの行数。
- **cols**: **number** OPTIONAL
  Sheetの列数。
- **pos**: **number** OPTIONAL
  sheetsコレクション内の位置。
- **grid**: **FlexGrid** OPTIONAL
  Sheet に関連付ける**FlexGrid** インスタンス。 これを指定しない場合、新しい**FlexGrid** インスタンスが作成されます。

戻り値 **Sheet**

applyCellsStyle

```
applyCellsStyle(cellStyle: ICellStyle, cells?: CellRange[], isPreview?: boolean): void
```

セルの範囲にスタイルを適用します。

パラメーター

- **cellStyle**: **ICellStyle**
  適用する**ICellStyle** オブジェクト。
- **cells**: **CellRange**[] OPTIONAL
  スタイルを適用する**CellRange** オブジェクトの配列。 これを指定しない場合、現在選択されているセルにスタイルが適用されます。
- **isPreview**: **boolean** OPTIONAL
  適用するスタイルがプレビュー用かどうかを示します。

戻り値 **void**

applyFunctionToCell

```
applyFunctionToCell(): void
```

関数リストで選択されている関数をセル値エディタに挿入します。

戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **className**s: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 `Control`
戻り値 **HTMLElement**

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

列をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **c**: **number**
  サイズ変更する列のインデックス。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 `FlexGrid`
戻り値 **void**

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

列の範囲をその内容がちょうど収まるようにサイズ変更します。

測定される行の範囲は常に、現在表示されているすべての行 + 現在表示されていない最大2,000行です。グリッドに大量のデータが含まれている場合には（たとえば、50,000行など）、すべての行を測定すると非常に時間がかかる可能性があるため、すべての行は測定されません。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstColumn**: **number** OPTIONAL
  サイズ変更する最初の列のインデックス（デフォルトは最初の列）。
- **lastColumn**: **number** OPTIONAL
  サイズ変更する最後の列のインデックス（デフォルトは最後の列）。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

行をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **r**: **number**
  サイズ変更する行のインデックス。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の列であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

行の範囲をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックス。
- **lastRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックスす。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の行であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

canEditCell

```
canEditCell(r: number, c: number): void
```

指定されたセルが編集可能かどうかを示す値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

継承元 **FlexGrid**
戻り値 **void**

clear

```
clear(): void
```

FlexSheetコントロールの内容をクリアします。

戻り値 **void**

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

すべてのグループ行を指定したレベルに折りたたみます。

パラメーター

- **level**: **number**
  表示する最大のグループレベル。

継承元 **FlexGrid**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

関数リストを考慮に入れて、基本クラスのメソッドをオーバーライドします。

戻り値 **boolean**

STATIC convertNumberToAlpha

```
convertNumberToAlpha(c: number): string
```

数値をその対応するアルファベットに変換します。 たとえば、0、1、2...をa、b、c...に変換します。

パラメーター

- **c**: **number**
  変換する数値。

戻り値 **string**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

deleteColumns

```
deleteColumns(index?: number, count?: number): void
```

FlexSheetコントロールの現在の**Sheet** から列を削除します。

パラメーター

- **index**: **number** OPTIONAL
  The starting index of the deleting columns. If not specified then columns will be deleted starting from the first column of the current selection.
- **count**: **number** OPTIONAL
  The numbers of columns to delete. If not specified then one column will be deleted.

戻り値 **void**

deleteRows

```
deleteRows(index?: number, count?: number): void
```

FlexSheetコントロールの現在の**Sheet** から行を削除します。

パラメーター

- **index**: **number** OPTIONAL
  削除する行の開始インデックス。 これを指定しない場合、現在選択されている範囲の最初の行から行が削除されます。
- **count**: **number** OPTIONAL
  削除する行数。 これを指定しない場合、1行が削除されます。

戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

evaluate

```
evaluate(formula: string, format?: string, sheet?: Sheet, getPrimitiveResult?: boolean): any
```

式を評価します。

FlexSheet formulas follow the Excel syntax, including a large subset of the functions supported by Excel. A complete list of the functions supported by **FlexSheet** can be found here: **FlexSheet Functions**.

パラメーター

- **formula**: **string**
  The formula to evaluate. The formula may start with an optional equals sign ('=').
- **format**: **string** OPTIONAL
  If specified, defines the .Net format that will be applied to the evaluated value.
- **sheet**: **Sheet** OPTIONAL
  The **Sheet** whose data will be used for evaluation. If not specified then the current sheet is used.
- **getPrimitiveResult**: **boolean** OPTIONAL
  Indicates whether need convert the non-primitive result to primitive type.

戻り値 **any**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

継承元 **FlexGrid**
戻り値 **boolean**

focus

```
focus(): void
```

グリッド全体をスクロールしてビューに入れることなくグリッドにフォーカスを設定するようにオーバーライドされます。

継承元 **FlexGrid**
戻り値 **void**

freezeAtCursor

```
freezeAtCursor(): void
```

FlexSheetコントロールの列および行を固定または固定解除します。

戻り値 **void**

getBuiltInTableStyle

```
getBuiltInTableStyle(styleName: string): TableStyle
```

Get the built-in table style via its name.

パラメーター

- **styleName**: **string**
  The name of the built-in table style.

戻り値 **TableStyle**

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セル要素の範囲（ビューポート座標単位）を取得します。

このメソッドは、**cells** パネル内のセル（スクロール可能なデータセル）の範囲を返します。その他のパネルにあるセルの範囲を取得するには、該当する**GridPanel** オブジェクトの**getCellBoundingRect** メソッドを使用してください。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。

- **c**: **number**
  セルを含む列のインデックス。

- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

継承元 **FlexGrid**
戻り値 **Rect**

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

グリッドのスクロール可能領域内のセルに格納された値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

継承元 **FlexGrid**
戻り値 **any**

getCellValue

```
getCellValue(rowIndex: number, colIndex: number, formatted?: boolean, sheet?: Sheet): any
```

評価されたセル値を取得します。

getCellDataメソッドは生のデータ（値または数式のどちらか）を返しますが、**getCellValue**メソッドはそれとは異なり、常に評価された値を返します。すなわち、セルに数式が含まれている場合は、まず数式を評価してから結果の値が返されます。

パラメーター

- **rowIndex**: **number**
  セルの行インデックス。
- **colIndex**: **number**
  セルの列インデックス。
- **formatted**: **boolean** OPTIONAL
  セルの元の値または書式設定された値のどちらを返すかを示します。
- **sheet**: **Sheet** OPTIONAL
  評価に使用されるデータを含む**Sheet**。 これを指定しない場合、現在のシートのデータが使用されます。

戻り値 **any**

getClipString

```
getClipString(rng?: CellRange): string
```

CellRange の内容を、クリップボードへのコピーに適した文字列として取得します。

FlexSheet は、このメソッドをオーバーライドして、 **FlexSheet** で複数の行または列の選択をサポートします。

非表示の行および列はクリップボード文字列に含まれません。

パラメーター

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

戻り値 **string**

getColumn

```
getColumn(name: string): Column
```

名前または連結に基づいて列を取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **string**
  検索する名前または連結。

継承元 **FlexGrid**
戻り値 **Column**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getMergedRange

```
getMergedRange(panel: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

GridPanel 内のセルの結合範囲を示す**CellRange** を取得します。 このメソッドは、親クラスであるFlexGridのgetMergedRangeメソッドをオーバーライドします。

パラメーター

- **panel**: **GridPanel**
  範囲を含む**GridPanel**。

- **r**: **number**
  セルを含む行のインデックス。

- **c**: **number**
  セルを含む列のインデックス。

- **clip**: **boolean** OPTIONAL
  結合範囲をグリッドの現在のビュー範囲にクリップするかどうか。

戻り値 **CellRange**

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

パラメーター

- **r**: **number**
  検査するセルの行インデックス。

- **c**: **number**
  検査するセルの列インデックス。

継承元 **FlexGrid**
戻り値 **SelectedState**

getSelectionFormatState

```
getSelectionFormatState(): IFormatState
```

選択されているセルの書式設定を記述する**IFormatState** オブジェクトを取得します。

戻り値 **IFormatState**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hideFunctionList

```
hideFunctionList(): void
```

関数リストを閉じます。

戻り値 **void**

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

指定されたポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

次に例を示します。

```
// ユーザーがグリッドをクリックしたときに、ポイントをヒットテストします
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

パラメーター

- **pt**: **any**
  調べる**Point** （ページ座標単位）、MouseEventオブジェクト、またはポイントのX座標。

- **y**: **any** OPTIONAL
  ポイントのY座標（ページ座標単位、最初のパラメーターが数値の場合）。

継承元 **FlexGrid**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

insertColumns

```
insertColumns(index?: number, count?: number): void
```

FlexSheetコントロールの現在の**Sheet** に列を挿入します。

パラメーター

- **index**: **number** OPTIONAL
  新しい列を追加する位置。 これを指定しない場合、現在選択されている範囲の一番左の列の前に列が追加されます。

- **count**: **number** OPTIONAL
  追加する列数。これを指定しない場合、1列が追加されます。

戻り値 **void**

insertRows

```
insertRows(index?: number, count?: number): void
```

FlexSheetコントロールの現在の**Sheet** に行を挿入します。

パラメーター

- **index**: **number** OPTIONAL
  新しい行を追加する位置。 これを指定しない場合、現在選択されている範囲の最初の行の前に行が追加されます。

- **count**: **number** OPTIONAL
  追加する行数。これを指定しない場合、1行が追加されます。

戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

指定したCellRangeがこのグリッドの行および列コレクションに対して有効かどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックする範囲。

継承元 **FlexGrid**
戻り値 **boolean**

load

```
load(workbook: any): void
```

ワークブックを**FlexSheet**にロードします。 このメソッドは、JSZip 2.5で動作します。

次に例を示します。

```
// このサンプルは、［ファイルを開く］ダイアログで選択したxlsxファイルを開き、
// FlexSheetにデータを挿入します。


// HTML
<input type="file"
    id="importFile"
    accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
/>
<div id="flexHost"></>


// JavaScript
var flexSheet = new wijmo.grid.FlexSheet("#flexHost"),
    importFile = document.getElementById('importFile');

importFile.addEventListener('change', function () {
    loadWorkbook();
});

function loadWorkbook() {
    var reader,
        file = importFile.files[0];
    if (file) {
        reader = new FileReader();
        reader.onload = function (e) {
            flexSheet.load(reader.result);
        };
        reader.readAsArrayBuffer(file);
    }
}
```

パラメーター

- **workbook**: **any**
  xlsxファイルコンテンツを含むWorkbookインスタンス、Blobインスタンス、base-64文字列、またはArrayBuffer。

戻り値 **void**

loadAsync

```
loadAsync(workbook: any, onLoaded?: (workbook: wijmo.xlsx.Workbook), onError?: (reason?: any)): void
```

ワークブックを**FlexSheet**に非同期にロードします。 このメソッドは、JSZip 3.0で動作します。

パラメーター

- **workbook**: **any**
  xlsxファイルコンテンツを含むワークブックインスタンス、 Blobインスタンス、base64文字列、またはArrayBuffer。
- **onLoaded**: **(workbook: wijmo.xlsx.Workbook)** OPTIONAL
  This callback provides access to the loaded workbook instance. Since this method is asynchronous, users cannot get the loaded workbook instance immediately. This callback has a single parameter, the loaded workbook instance.
- **onError**: **(reason?: any)** OPTIONAL
  This callback catches errors when loading workbooks. It has a single parameter, the failure reason.

  For example:

```
flexsheet.loadAsync(blob, function (workbook) {

     // user can access the loaded workbook instance in this callback.
     var app = worksheet.application ;
     ...
}, function (reason) {

     // User can catch the failure reason in this callback.
     console.log('The reason of load failure is ' + reason);
});
```

戻り値 **void**

mergeRange

```
mergeRange(cells?: CellRange, isCopyMergeCell?: boolean): void
```

選択されている**CellRange** を1つのセルに結合します。

パラメーター

- **cells**: **CellRange** OPTIONAL
  結合する**CellRange**。
- **isCopyMergeCell**: **boolean** OPTIONAL
  このパラメータは、結合操作が結合セルのコピー/ペーストで行われたかどうかを示します。

戻り値 **void**

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

autoSizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

autoSizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

autoSizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

autoSizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

beginningEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

cellEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

cellEditEnding イベントを発生させます。

パラメーター

- **e**: **CellEditEndingEventArgs**
  イベントデータを含む**CellEditEndingEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onColumnChanged

```
onColumnChanged(e: RowColumnChangedEventArgs): void
```

columnChangedイベントを発生させます。

パラメーター

- **e**: **RowColumnChangedEventArgs**

戻り値 **void**

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

copied イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

copying イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

deletedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

deletingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

draggedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

draggedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

draggingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

draggingColumnOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

draggingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRowColumn

```
onDraggingRowColumn(e: DraggingRowColumnEventArgs): void
```

draggingRowColumnイベントを発生させます。

パラメーター

- **e**: **DraggingRowColumnEventArgs**

戻り値 **void**

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

draggingRowOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDroppingRowColumn

```
onDroppingRowColumn(e?: EventArgs): void
```

droppingRowColumnイベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

groupCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

groupCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onItemsSourceChanging

```
onItemsSourceChanging(e: CancelEventArgs): boolean
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLoaded

```
onLoaded(e?: EventArgs): void
```

loadedイベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

loadedRows イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

loadingRows イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

pasted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

pastedCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

pasting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

pastingCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

prepareCellForEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPrepareChangingColumn

```
onPrepareChangingColumn(e: RowColumnChangedEventArgs): void
```

prepareChangingColumnイベントを発生させます。

パラメーター

- **e**: **RowColumnChangedEventArgs**

戻り値 **void**

onPrepareChangingRow

```
onPrepareChangingRow(e: RowColumnChangedEventArgs): void
```

prepareChangingRowイベントを発生させます。

パラメーター

- **e**: **RowColumnChangedEventArgs**

戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

resizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

resizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

resizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

resizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

rowAdded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowChanged

```
onRowChanged(e: RowColumnChangedEventArgs): void
```

rowChangedイベントを発生します。

パラメーター

- **e**: **RowColumnChangedEventArgs**

戻り値 **void**

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

rowEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

rowEditEnding イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

rowEditStarted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

rowEditStarting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

scrollPositionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onSelectedSheetChanged

```
onSelectedSheetChanged(e: PropertyChangedEventArgs): void
```

currentSheetChangedイベントを発生させます。

パラメーター

- **e**: **PropertyChangedEventArgs**
  イベントデータを含む**PropertyChangedEventArgs**。

戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

selectionChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onSheetCleared

```
onSheetCleared(e?: EventArgs): void
```

sheetClearedイベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

sortedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

sortingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUnknownFunction

```
onUnknownFunction(e: UnknownFunctionEventArgs): void
```

unknownFunctionイベントを発生させます。

パラメーター

- **e**: **UnknownFunctionEventArgs**

戻り値 **void**

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

updatedLayout イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

updatingLayout イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

updatingView イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

redo

```
redo(): void
```

最後のユーザー操作をやり直します。

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

シートとTabHolderをリフレッシュするためにオーバーライドされます。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean**
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

- **recycle**: **boolean** OPTIONAL
  既存の要素を再利用するかどうか。

- **state**: **boolean** OPTIONAL
  既存の要素を保持してその状態を更新するかどうか。

継承元 **FlexGrid**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

save

```
save(fileName?: string): Workbook
```

FlexSheetをxlsxファイルに保存します。 このメソッドは、JSZip 2.5で動作します。

次に例を示します。

```
// このサンプルは、ボタンのクリックで、FlexSheetのコンテンツをxlsxにエクスポートします。
// click.


// HTML
<button
    onclick="saveXlsx('FlexSheet.xlsx')">
    Save
</button>


// JavaScript
function saveXlsx(fileName) {

    // flexGridをxlsxファイルに保存します。
    flexsheet.save(fileName);
}
```

パラメーター

- **fileName**: **string** OPTIONAL
  生成されるファイル名。

戻り値 **Workbook**

saveAsync

```
saveAsync(fileName?: string, onSaved?: (base64?: string), onError?: (reason?: any)): void
```

FlexSheetをxlsxファイルに非同期に保存します。 このメソッドは、JSZip 3.0で動作します。

パラメーター

- **fileName**: **string** OPTIONAL
  生成されるファイル名。

- **onSaved**: **(base64?: string)** OPTIONAL
  このコールバックは、保存されたFlexSheetのコンテンツを表すbase64文字列を取得する方法を提供します。 このメソッドは非同期メソッドであるため、 ユーザーはbase64文字列を直ちに取得することができません。 このコールバックを通じて base64文字列を取得する必要があります。 コールバックのパラメータには、保存されたflexsheetのbase64文字列が含まれます。 それがユーザーに渡されます。

- **onError**: **(reason?: any)** OPTIONAL
  このコールバックは、保存時のエラー情報を捕捉します。 コールバックのパラメータには、失敗理由が含まれます。 保存失敗の理由を確認したい場合は、 この戻り値がユーザーに渡されます。

  次に例を示します。

```
flexsheet.saveAsync('', function (base64) {

    // このコールバックで、ユーザーはbase64文字列にアクセスできます。
    document.getElementById('export').href = 'data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;' + 'base64
,' + base64;
}, function (reason) {

    // このコールバックで、ユーザーは失敗理由を確認できます。
    console.log('The reason of save failure is ' + reason);
});
```

- **options**: **IFlexSheetXlsxOptions** OPTIONAL
  `IFlexSheetXlsxOptions` object specifying the save options.

戻り値 **void**

scrollIntoView

```
scrollIntoView(r: number, c: number, refresh?: boolean): boolean
```

指定したセルが画面に入るようにグリッドをスクロールします。

負の行と列のインデックスは無視されるので、以下を呼び出すと、

```
grid.scrollIntoView(200, -1);
```

グリッドは200行目を表示するように垂直にスクロールしますが、水平にスクロールしません。

パラメーター

- **r**: **number**
  画面に入るようにスクロールする行のインデックス

- **c**: **number**
  画面に入るようにスクロールする列のインデックス。

- **refresh**: **boolean** OPTIONAL
  スクロールした新しい位置をすぐ表示するためにグリッドを更新する必要があるかどうかを決定するオプションのパラメータ。

継承元 **`FlexGrid`**
戻り値 **boolean**

select

```
select(rng: any, show?: any): void
```

セル範囲を選択し、オプションでそのセル範囲が画面に入るようにスクロールします。

FlexSheet はこのメソッドをオーバーライドして、 **FlexSheet** 上の結合されたセルに合わせて選択セル範囲を調整します。

パラメーター

- **rng**: **any**
  選択するセル範囲。
- **show**: **any** OPTIONAL
  新しい選択範囲が画面に入るようにスクロールするかどうかを示します。

戻り値 **void**

selectNextFunction

```
selectNextFunction(): void
```

関数リスト内の次の関数を選択します。

戻り値 **void**

selectPreviousFunction

```
selectPreviousFunction(): void
```

関数リスト内の前の関数を選択します。

戻り値 **void**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

基本クラスのsetCellData関数をオーバーライドします。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **value**: **any**
  セルに格納する値。
- **coerce**: **boolean** OPTIONAL
  列のデータ型に合わせて値を自動的に変更するかどうか。
- **invalidate**: **boolean** OPTIONAL
  FlexSheetを無効にして変更を表示するかどうか。

戻り値 **boolean**

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

文字列を行および列に解析し、その内容を特定の範囲に適用します。

FlexGrid の**setClipString**メソッドをオーバーライドします。

パラメーター

- **text**: **string**
  グリッドに解析する、タブと改行で区切られたテキスト。

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

戻り値 **void**

showColumnFilter

```
showColumnFilter(): void
```

フィルタエディタを表示します。

戻り値 **void**

showFunctionList

```
showFunctionList(target: HTMLElement): void
```

関数リストを開きます。

パラメーター

- **target**: **HTMLElement**
  関数リストの表示/非表示を切り替えるDOM要素。

戻り値 **void**

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

指定されたセルの編集を開始します。

FlexGrid の編集は、Excelの編集によく似ています。 ［F2］キーを押すか、セルをダブルクリックすると、 グリッドが**完全編集** モードになります。 このモードでは、ユーザーが［Enter］、［Tab］、または［Esc］キーを押すか、マウスを使用して選択範囲を移動するまで、セルエディタはアクティブのままになります。完全編集モードでは、カーソルキーを押しても、グリッドの編集モードは終了しません。

テキストをセルに直接入力すると、グリッドは**クイック編集**モードになります。 このモードでは、ユーザーがEnter、Tab、Esc、またはいずれかの矢印キーを押すまで、セルエディタはアクティブなままになります。

通常、完全編集モードは既存の値を変更する際に使用します。 クイック編集モードは新しいデータをすばやく入力する際に使用します。

編集中に［F2］キーを押すことで、完全編集モードとクイック編集モードを切り替えることができます。

パラメーター

- **fullEdit**: **boolean** OPTIONAL
  ユーザーがカーソルキーを押したときも編集モードのままにするかどうか。 デフォルトはtrueです。
- **r**: **number** OPTIONAL
  編集する行のインデックス。 デフォルトは現在選択されている行です。
- **c**: **number** OPTIONAL
  編集する列のインデックス。 デフォルトは現在選択されている列です。
- **focus**: **boolean** OPTIONAL
  編集開始時に、エディタにフォーカスを与えるかどうか。 デフォルトはtrueです。

継承元 **FlexGrid**
戻り値 **boolean**

toggleDropDownList

```
toggleDropDownList(): void
```

現在選択されているセルに関連付けられたドロップダウンリストボックスを切り替えます。

このメソッドでは、セルが編集モードに入ったとき、またはユーザが特定のキーを押したときに、ドロップダウンリストを自動的に表示することができます。

たとえば、このコードでは、グリッドが編集モードに入るたびに、グリッドにドロップダウンリストが表示されます。

```
// グリッドが編集モードに入ると、ドロップダウンリストを表示する。
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

このコードでは、ユーザーがスペースバーを押した場合、グリッドが編集モードに入った後で、ドロップダウンリストが表示されます。

```
// ユーザーがスペースバーを押したときにドロップダウンリストを表示する。
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

継承元 **FlexGrid**
戻り値 **void**

undo

```
undo(): void
```

最後のユーザー操作を元に戻します。

戻り値 **void**

イベント

autoSizedColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizedRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

beginningEdit

セルが編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnded

セル編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnding

セル編集が終了するときに発生します。

このイベントを使用して検証を実行し、無効な編集を防ぐことができます。 たとえば、次のコードは、ユーザーが文字「a」を含まない値を入力できないようにします。このコードは、編集が適用される前に、編集前と編集後の値を取得する方法を示しています。

```
function cellEditEnding (sender, e) {

  // 編集前と編集後の値を取得します
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // newValに「a」が含まれていない場合は、編集をキャンセルします
  e.cancel = newVal.indexOf('a') < 0;
}
```

cancel パラメータをtrueに設定すると、編集された値が無視されて、グリッドでセルの元の値が維持されます。

また、**stayInEditMode** パラメータをtrueに設定すると、グリッドの編集モードが維持され、編集をコミットする前にユーザーが無効な値を修正できます。

継承元 **FlexGrid**
引数 **CellEditEndingEventArgs**

columnChanged

FlexSheet で列が挿入/削除された後に発生します。

引数 **RowColumnChangedEventArgs**

copied

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーした後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

copying

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーするときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラでコピー操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletedRow

ユーザーが［Del］キーを押して行を削除した後に発生します （**allowDelete** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletingRow

ユーザーが［Delete］キーを押して選択されている行を削除するときに発生します（**allowDelete** プロパティを参照）。

イベントハンドラで行の削除をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedColumn

ユーザーが列のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedRow

ユーザーが行のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumn

ユーザーが列のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumnOver

ユーザーが列を別の位置にドラッグするときに発生します。

ハンドラは、このイベントをキャンセルして、ユーザーが特定の位置に列をドロップしないようにすることができます。次に例を示します。

```
// ドラッグされている列を記憶します
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// 'sales'列がインデックス0にドラッグされないようにします
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRow

ユーザーが行のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRowColumn

FlexSheetの行または列をドラッグしているときに発生します。

引数　　**DraggingRowColumnEventArgs**

draggingRowOver

ユーザーが行を別の位置にドラッグするときに発生します。

継承元　　**FlexGrid**
引数　　**CellRangeEventArgs**

droppingRowColumn

FlexSheetの行または列をドロップしているときに発生します。

引数　　**EventArgs**

formatItem

セルを表す要素が作成されたときに発生します。

このイベントを使用してセルを表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、グループ行のセルから'wj-wrap'クラスを削除します。

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

継承元　　**FlexGrid**
引数　　**FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元　　**Control**
引数　　**EventArgs**

groupCollapsedChanged

グループが展開または折りたたまれた後に発生します。

継承元　　**FlexGrid**
引数　　**CellRangeEventArgs**

groupCollapsedChanging

グループが展開または折りたたまれる直前に発生します。

継承元　　**FlexGrid**
引数　　**CellRangeEventArgs**

itemsSourceChanged

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

itemsSourceChanging

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

loaded

FlexSheet が**Workbook** インスタンスをロードした後に発生します。

引数 **EventArgs**

loadedRows

グリッド行がデータソースの項目に連結された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

loadingRows

グリッド行がデータソースの項目に連結される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

pasted

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastedCell

ユーザーがクリップボードの内容をセルに貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pasting

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた時に発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastingCell

ユーザーがクリップボードの内容をセルに貼り付けるときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

prepareCellForEdit

エディタセルが作成されたとき、それがアクティブになる前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

prepareChangingColumn

FlexSheet で列が挿入/削除される前に発生します。

引数 **RowColumnChangedEventArgs**

prepareChangingRow

FlexSheet で行が挿入/削除される前に発生します。

引数 **RowColumnChangedEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

resizedColumn

ユーザーが列のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizedRow

ユーザーが行のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingColumn

列がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingRow

行がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowAdded

ユーザーが新規行テンプレートを編集して新しい項目を作成したときに発生します（**allowAddNew** プロパティを参照）。

イベントハンドラで新しい項目の内容をカスタマイズしたり、新しい項目の作成をキャンセルしたりできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowChanged

FlexSheet で行が挿入/削除された後に発生します。

引数 **RowColumnChangedEventArgs**

rowEditEnded

行編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnding

行編集が終了するとき、変更が確定またはキャンセルされる前に発生します。

このイベントを**rowEditStarted**イベントと組み合わせて使用して、ディープ連結編集を元に戻す操作を実装できます。次に例を示します。

```
// 編集の開始時にディープ連結値を保存します
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.')> -1) { // ディープ連結
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// 編集がキャンセルされたときにディープ連結値を復元します
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // ユーザーによって編集がキャンセルされました
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarted

行が編集モードに入った後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarting

行が編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

scrollPositionChanged

コントロールがスクロールされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

selectedSheetChanged

現在のシートインデックスが変更されたときに発生します。

引数 **PropertyChangedEventArgs**

selectionChanged

選択が変更された後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

selectionChanging

選択が変更される前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sheetCleared

FlexSheet がクリアされたときに発生します。

引数 **EventArgs**

sortedColumn

ユーザーが列ヘッダをクリックしてソートを適用した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortingColumn

ユーザーが列ヘッダをクリックしてソートを適用する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

unknownFunction

FlexSheet で不明な式が検出されたときに発生します。

引数 **UnknownFunctionEventArgs**

updatedLayout

グリッドで内部レイアウトが更新された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

updatedView

グリッドが現在のビューを構成する要素の作成/更新を完了したときに発生します。

グリッドのビューは以下のような操作に応じて更新されます。

- グリッドまたはそのデータソースの更新
- 行または列の追加、削除、変更
- グリッドのサイズ変更またはスクロール
- 選択の変更

継承元 **FlexGrid**
引数 **EventArgs**

updatingLayout

グリッドで内部レイアウトが更新される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

updatingView

現在のビューを構成する要素の作成/更新をグリッドが開始したときに発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

FlexSheetColumnFilter クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **ColumnFilter**
表示 ☐継承されたメンバー ☐イベント発生元

FlexSheet コントロールの列のフィルタを定義します。

FlexSheetColumnFilter には**FlexSheetConditionFilter** と**FlexSheetValueFilter** が含まれます。一度にアクティブにできるのはどちらか一方だけです。

このクラスは**FlexSheetFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- column
- conditionFilter
- dataMap
- filterType
- isActive
- valueFilter

メソッド

- apply
- clear
- implementsInterface

コンストラクタ

constructor

```
constructor(owner: FlexSheetFilter, column: Column): FlexSheetColumnFilter
```

FlexSheetColumnFilter クラスの新しいインスタンスを初期化します。

パラメーター

- **owner**: **FlexSheetFilter**
  この列フィルタを所有する**FlexSheetFilter** 。
- **column**: **Column**
  フィルタリングする**Column** 。

戻り値 **FlexSheetColumnFilter**

プロパティ

column

フィルタリングする**Column** を取得します。

継承元 **ColumnFilter**
型 **Column**

conditionFilter

このColumnFilter のConditionFilter を取得します。

継承元 ColumnFilter
型 ConditionFilter

dataMap

未加工の値をこのフィルタを編集する際に表示される表示値に変換するために使用されるDataMap を取得または設定します。

次の例では、DataMap をBoolean型の列フィルタに割り当て、 フィルタエディタに、'true'と'false'ではなく'Yes'と'No'が表示されるようにしています。

```
var filter = new wijmo.grid.filter.FlexGridFilter(grid),
    map = new wijmo.grid.DataMap([
            { value: true, caption: 'Yes' },
            { value: false, caption: 'No' },
        ], 'value', 'caption');
for (var c = 0; c < grid.columns.length; c++) {
    if (grid.columns[c].dataType == wijmo.DataType.Boolean) {
        filter.getColumnFilter(c).dataMap = map;
    }
}
```

継承元 ColumnFilter
型 DataMap

filterType

このフィルタから提供されるフィルタ処理のタイプを取得または設定します。

このプロパティをnullに設定すると、フィルタは、オーナーフィルタの defaultFilterType プロパティで定義された値を使用します。

継承元 ColumnFilter
型 FilterType

isActive

このフィルタがアクティブかどうかを示す値を取得します。

継承元 ColumnFilter
型 boolean

valueFilter

このColumnFilter のValueFilter を取得します。

継承元 ColumnFilter
型 ValueFilter

メソッド

apply

```
apply(value): boolean
```

値がフィルタに一致するかどうかを示す値を取得します。

パラメーター

- **value**:
  テストする値。

継承元 **ColumnFilter**
戻り値 **boolean**

clear

```
clear(): void
```

フィルタをクリアします。

継承元 **ColumnFilter**
戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **ColumnFilter**
戻り値 **boolean**

FlexSheetColumnFilterEditor クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **ColumnFilterEditor**
表示 ☐継承されたメンバー ☐イベント発生元

列フィルタの設定の確認や変更に使用されるエディタ。

このクラスは**FlexSheetFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onButtonClicked
- onFilterChanged
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

イベント

- buttonClicked
- filterChanged
- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, filter: FlexSheetColumnFilter, sortButtons?: boolean): FlexSheetColumnFilterEditor
```

FlexSheetColumnFilterEditor クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **filter**: **FlexSheetColumnFilter**
  編集する**FlexSheetColumnFilter** 。
- **sortButtons**: **boolean** OPTIONAL
  エディタにソートボタンを表示するかどうか。

戻り値 **FlexSheetColumnFilterEditor**

プロパティ

STATIC controlTemplate

ColumnFilterEditor コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **ColumnFilterEditor**
型 **any**

filter

編集する**ColumnFilter** への参照を取得します。

継承元 **ColumnFilterEditor**
型 **ColumnFilter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onButtonClicked

```
onButtonClicked(e?: EventArgs): void
```

buttonClicked イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ColumnFilterEditor**
戻り値 **void**

onFilterChanged

```
onFilterChanged(e?: EventArgs): void
```

filterChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ColumnFilterEditor**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

updateEditor

```
updateEditor(): void
```

現在のフィルタ設定でエディタを更新します。

継承元 **ColumnFilterEditor**
戻り値 **void**

updateFilter

```
updateFilter(): void
```

現在のエディタ設定でフィルタを更新します。

継承元 **ColumnFilterEditor**
戻り値 **void**

イベント

buttonClicked

いずれかのエディタボタンがクリックされたときに発生します。

継承元 **ColumnFilterEditor**
引数 **EventArgs**

filterChanged

フィルタが変更された後に発生します。

継承元 **ColumnFilterEditor**
引数 **EventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

FlexSheetConditionFilter クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **ConditionFilter**
表示 ☐継承されたメンバー ☐イベント発生元

FlexSheet コントロールの列の条件フィルタを定義します。

条件フィルタには、'and'または'or'演算子を使用して結合できる2つの条件が含まれます。

このクラスは**FlexSheetFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- and
- column
- condition1
- condition2
- dataMap
- isActive

メソッド

- apply
- clear
- implementsInterface

コンストラクタ

constructor

```
constructor(column: Column): FlexSheetConditionFilter
```

ConditionFilter クラスの新しいインスタンスを初期化します。

パラメーター
- **column**: **Column**
  フィルタリングする列。

戻り値 **FlexSheetConditionFilter**

プロパティ

- and

2つの条件をAND演算子とOR演算子のどちらを使用して結合するかを示す値を取得します。

継承元 **ConditionFilter**
型 **boolean**

column

フィルタリングする**Column** を取得します。

継承元 **ConditionFilter**
型 **Column**

condition1

このフィルタの最初の条件を取得します。

継承元 **ConditionFilter**
型 **FilterCondition**

condition2

このフィルタの2番目の条件を取得します。

継承元 **ConditionFilter**
型 **FilterCondition**

dataMap

未加工の値をこのフィルタを編集する際に表示される表示値に変換するために使用される**DataMap** を取得または設定します。

継承元 **ConditionFilter**
型 **DataMap**

isActive

このフィルタがアクティブかどうかを示す値を取得します。

2つの条件のうち少なくとも1つの条件の演算子と値が有効な組み合わせに設定されている場合、そのフィルタはアクティブです。

継承元 **ConditionFilter**
型 **boolean**

メソッド

apply

```
apply(value): boolean
```

値がこのフィルタに一致するかどうかを示す値を返します。

パラメーター

- **value**:
  テストする値。

戻り値 **boolean**

clear

```
clear(): void
```

フィルタをクリアします。

継承元 **ConditionFilter**
戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **ConditionFilter**
戻り値 **boolean**

FlexSheetFilter クラス

ファイル	`wijmo.grid.sheet.js`
モジュール	**wijmo.grid.sheet**
基本クラス	**FlexGridFilter**
表示	☐ 継承されたメンバー ☐ イベント発生元

FlexSheet コントロールに対してExcelスタイルのフィルタを実装します。

FlexSheet コントロールでフィルタリングを有効にするには、**FlexSheetFilter** のインスタンスを作成し、コンストラクターのパラメーターとしてグリッドを渡します。

コンストラクタ

- constructor

プロパティ

- activeEditor
- defaultFilterType
- filterColumns
- filterDefinition
- grid
- showFilterIcons
- showSortButtons

メソッド

- apply
- clear
- closeEditor
- editColumnFilter
- getColumnFilter
- onFilterApplied
- onFilterChanged
- onFilterChanging

イベント

- filterApplied
- filterChanged
- filterChanging

コンストラクタ

constructor

```
constructor(grid: FlexGrid, options?: any): FlexGridFilter
```

FlexGridFilter クラスの新しいインスタンスを初期化します。

パラメーター

- **grid**: **FlexGrid**
  フィルタ対象の**FlexGrid** 。
- **options**: **any** OPTIONAL
  FlexGridFilter の初期化オプション。

継承元 **FlexGridFilter**
戻り値 **FlexGridFilter**

プロパティ

activeEditor

アクティブな**ColumnFilterEditor** を取得します。

このプロパティを使用すると、**filterChanging** イベントを処理するときにフィルターエディターをカスタマイズできます。 フィルターが編集されていない場合はnullを返します。

継承元 **FlexGridFilter**
型 **ColumnFilterEditor**

defaultFilterType

使用するデフォルトフィルタタイプを取得または設定します。

この値は特定の列のフィルタでオーバーライドできます。 たとえば、以下のサンプルコードは、"ByValue"列を除くすべての列に対して条件に基づくフィルタを作成します。

```
var f = new wijmo.grid.filter.FlexGridFilter(flex);
f.defaultFilterType = wijmo.grid.filter.FilterType.Condition;
var col = flex.columns.getColumn('ByValue'),
    cf = f.getColumnFilter(col);
cf.filterType = wijmo.grid.filter.FilterType.Value;
```

The default value for this property is **FilterType.Both**.

継承元 **FlexGridFilter**
型 **FilterType**

filterColumns

フィルタを持つ列の名前またはバインディングを含む配列を取得または設定します。

このプロパティをnullまたは空の配列に設定すると、すべての列にフィルタが追加されます。

継承元 **FlexGridFilter**
型 **string**[]

filterDefinition

現在のフィルタ定義をJSON文字列として取得または設定します。

型 **string**

grid

このフィルタを所有する**FlexGrid** への参照を取得します。

継承元 **FlexGridFilter**
型 **FlexGrid**

showFilterIcons

FlexGridFilter がグリッドの列ヘッダにフィルタ編集ボタンを追加するかどうかを示す値を取得または設定します。

このプロパティをfalseに設定した場合は、ユーザーがフィルタを編集、クリア、および適用する手段を開発者が提供する必要があります。

継承元 **FlexGridFilter**
型 **boolean**

showSortButtons

フィルタエディタにソートボタンが表示されるかどうかを示す値を取得または設定します。

デフォルトでは、エディタにはExcelと同じようにソートボタンが表示されます。しかし、ユーザーはヘッダをクリックすることによって列をソートできるので、 フィルタエディタにソートボタンがあるのは望ましくない場合があります。

The default value for this property is **true**.

継承元 **FlexGridFilter**
型 **boolean**

メソッド

apply

```
apply(): void
```

現在の列フィルタをシートに適用します。

戻り値 **void**

clear

```
clear(): void
```

すべての列フィルタをクリアします。

継承元 **FlexGridFilter**
戻り値 **void**

closeEditor

```
closeEditor(): void
```

フィルタエディタを閉じます。

戻り値 **void**

editColumnFilter

```
editColumnFilter(col: any, ht?: HitTestInfo): void
```

指定したグリッド列のフィルタエディタを表示します。

パラメーター

- **col**: **any**
  編集するフィルタを含む**Column** 。
- **ht**: **HitTestInfo** OPTIONAL
  フィルタ表示をトリガしたセルの範囲を含む**HitTestInfo** オブジェクト。

戻り値 **void**

getColumnFilter

```
getColumnFilter(col: any, create?: boolean): FlexSheetColumnFilter
```

指定した列のフィルタを取得します。

パラメーター

- **col**: **any**
  The **Column** that the filter applies to (or column name or index).
- **create**: **boolean** OPTIONAL
  Whether to create the filter if it does not exist.

戻り値 **FlexSheetColumnFilter**

onFilterApplied

```
onFilterApplied(e?: EventArgs): void
```

filterApplied イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGridFilter**
戻り値 **void**

onFilterChanged

```
onFilterChanged(e: CellRangeEventArgs): void
```

filterChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**

継承元 **FlexGridFilter**
戻り値 **void**

onFilterChanging

```
onFilterChanging(e: CellRangeEventArgs): boolean
```

filterChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGridFilter**
戻り値 **boolean**

イベント

filterApplied

フィルタが適用された後に発生します。

継承元 **FlexGridFilter**
引数 **EventArgs**

filterChanged

ユーザーが列フィルタを編集した後で発生します。

イベントパラメータを使用して、フィルタを所有する列を判定し、 変更が適用されたかキャンセルされたかを判定します。

継承元 **FlexGridFilter**
引数 **CellRangeEventArgs**

filterChanging

ユーザーが列フィルタを編集しようとしたときに発生します。

フィルタのデフォルトの設定をオーバーライドする場合は、 このイベントを使用して列フィルタをカスタマイズします。

たとえば、以下のコードは、フィルタ条件がnullの場合に、使用される演算子を 'contains'に設定します。

```
filter.filterChanging.addHandler(function (s, e) {
  var cf = filter.getColumnFilter(e.col);
  if (!cf.valueFilter.isActive && cf.conditionFilter.condition1.operator == null) {
    cf.filterType = wijmo.grid.filter.FilterType.Condition;
    cf.conditionFilter.condition1.operator = wijmo.grid.filter.Operator.CT;
  }
});
```

継承元 **FlexGridFilter**
引数 **CellRangeEventArgs**

FlexSheetPanel クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **GridPanel**
表示 ☐継承されたメンバー ☐イベント発生元

GridPanel クラスの拡張クラスを定義します。これは、その基本クラスである **FlexGrid** が**GridPanel** を使用するのと同じように、**FlexSheet**によって使用されます。たとえば、**cells**プロパティはこのクラスのインスタンスを返します。

コンストラクタ

- constructor

プロパティ

- cellType
- columns
- grid
- height
- hostElement
- rows
- viewRange
- width

メソッド

- getCellBoundingRect
- getCellData
- getCellElement
- getSelectedState
- setCellData

コンストラクタ

constructor

```
constructor(grid: FlexGrid, cellType: CellType, rows: RowCollection, cols: ColumnCollection, element: HTMLElement): FlexSheetPanel
```

FlexSheetPanel クラスの新しいインスタンスを初期化します。

パラメーター

- **grid**: **FlexGrid**
  パネルを所有する**FlexGrid** オブジェクト。
- **cellType**: **CellType**
  パネル内のセルのタイプ。
- **rows**: **RowCollection**
  パネルに表示される行。
- **cols**: **ColumnCollection**
  パネルに表示される列。
- **element**: **HTMLElement**
  コントロール内のセルをホストするHTMLElement。

戻り値 **FlexSheetPanel**

プロパティ

cellType

パネルに含まれるセルのタイプを取得します。

継承元 **GridPanel**
型 **CellType**

columns

パネルの列コレクションを取得します。

継承元 **GridPanel**
型 **ColumnCollection**

grid

パネルを所有するグリッドを取得します。

継承元 **GridPanel**
型 **FlexGrid**

height

このパネルに含まれる内容全体の高さを取得します。

継承元 **GridPanel**
型 **number**

hostElement

パネルのホスト要素を取得します。

継承元 **GridPanel**
型 **HTMLElement**

rows

パネルの行コレクションを取得します。

継承元 **GridPanel**
型 **RowCollection**

viewRange

このパネル上の現在表示されているセルの範囲を示す**CellRange** を取得します。

継承元 **GridPanel**
型 **CellRange**

width

パネルに含まれる内容全体の幅を取得します。

継承元 **GridPanel**
型 **number**

メソッド

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セルの範囲（ビューポート座標単位）を取得します。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

継承元 **GridPanel**
戻り値 **Rect**

getCellData

```
getCellData(r: number, c: any, formatted: boolean): any
```

パネル内のセルに格納されている値を取得します。

パラメーター

- **r**: **number**
  セルの行インデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

戻り値 **any**

getCellElement

```
getCellElement(r: number, c: number): HTMLElement
```

この**GridPanel** 内のセルを表す要素を取得します。

セルが現在表示されていない場合、このメソッドはnullを返します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

継承元 **GridPanel**
戻り値 **HTMLElement**

getSelectedState

```
getSelectedState(r: number, c: number, rng: CellRange): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

このメソッドをオーバーライドすると、**FlexSheet** で複数範囲の選択されたヘッダの表示をサポートできます。

パラメーター

- **r**: **number**
  セルの行インデックスを指定します。
- **c**: **number**
  セルの列インデックスを指定します。
- **rng**: **CellRange**
  対象のセルを含む**CellRange**。

戻り値 **SelectedState**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

パネル内のセルの内容を設定します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **value**: **any**
  セルに格納する値。
- **coerce**: **boolean** OPTIONAL
  列のデータ型に合わせて値を自動的に変更するかどうかを示す値。
- **invalidate**: **boolean** OPTIONAL
  FlexSheetを無効にして変更を表示するかどうか。

戻り値 **boolean**

FlexSheetValueFilter クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **ValueFilter**
表示 ☐継承されたメンバー ☐イベント発生元

FlexSheet コントロールの列の値フィルタを定義します。

値フィルタには、シートに表示する値の明示的なリストが含まれます。

コンストラクタ

- constructor

プロパティ

- column
- dataMap
- filterText
- isActive
- maxValues
- showValues
- sortValues
- uniqueValues

メソッド

- apply
- clear
- implementsInterface

コンストラクタ

constructor

```
constructor(column: Column): FlexSheetValueFilter
```

FlexSheetValueFilter クラスの新しいインスタンスを初期化します。

パラメーター
- **column**: **Column**
  フィルタリングする列。

戻り値 **FlexSheetValueFilter**

プロパティ

- column

フィルタリングする**Column** を取得します。

継承元 **ValueFilter**
型 **Column**

dataMap

未加工の値をこのフィルタを編集する際に表示される表示値に変換するために使用される**DataMap** を取得または設定します。

継承元 **ValueFilter**
型 **DataMap**

filterText

表示値のリストのフィルタリングに使用される文字列を取得または設定します。

継承元 **ValueFilter**
型 **string**

isActive

このフィルタがアクティブかどうかを示す値を取得します。

少なくとも1つの値が選択されている場合、そのフィルタはアクティブです。

継承元 **ValueFilter**
型 **boolean**

maxValues

表示値のリスト内にある要素の最大数を取得または設定します。

非常に多くの項目をリストに追加すると、検索が困難になり、パフォーマンスが損なわれます。このプロパティは任意の時点で表示される項目数を制限しますが、 ユーザーは検索ボックスを使用して目的の項目をフィルタ処理することができます。 このプロパティは、デフォルトでは250に設定されます。

次のコードは、この値を1,000,000に変更し、事実上、フィールドのすべての一意の値を一覧します。

```
// 'id'列のmaxItemsプロパティを変更します。
var f = new wijmo.grid.filter.FlexGridFilter(s);
f.getColumnFilter('id').valueFilter.maxValues = 1000000;
```

継承元 **ValueFilter**
型 **number**

showValues

値リストに表示されるすべての書式設定された値を含むオブジェクトを取得または設定します。

継承元 **ValueFilter**
型 **any**

sortValues

エディタに表示するときに値をソートするかどうかを決定する値を取得または設定します。

uniqueValues を使用して値のカスタムリストを提供した上、 それらの値の順序を維持したい場合、このプロパティは特に便利です。

継承元 **ValueFilter**
型 **boolean**

uniqueValues

リストに表示する一意の値を含む配列を取得または設定します。

このプロパティがnullに設定されている場合、リストには、グリッドデータに基づいて値が挿入されます。

データからリストを作成するより、一意の値のリストを明示的に割り当てる方が 効率的です。また、データがサーバー上でフィルタ処理される場合、値フィルタが 適切に動作するには、そうする必要があります（この場合、一部の値が クライアント上に存在しない可能性があり、リストが不完全になるため）。

デフォルトでは、フィルタエディタは、一意の値をユーザーに表示する際に値をソートします。この動作を抑止して、指定した順序で値を表示する場合は、 **sortValues** プロパティをfalseに設定します。

たとえば、次のコードは、**ValueFilter** で'country'フィールドに連結された列に 使用される国名リストを提供します。

```
// FlexGridのフィルタを作成します
var filter = new wijmo.grid.filter.FlexGridFilter(grid);

// 一意の値のリストを国フィルタに割り当てます
var cf = filter.getColumnFilter('country');
cf.valueFilter.uniqueValues = countries;
```

継承元 **ValueFilter**
型 **any**[]

メソッド

apply

```
apply(value): boolean
```

値がフィルタに合致するかどうかを示す値を取得します。

パラメーター

- **value**:
  テストする値。

戻り値 **boolean**

clear

```
clear(): void
```

フィルタをクリアします。

継承元 **ValueFilter**
戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元	**`ValueFilter`**
戻り値	**boolean**

FlexSheetValueFilterEditor クラス

ファイル	`wijmo.grid.sheet.js`
モジュール	**wijmo.grid.sheet**
基本クラス	**ValueFilterEditor**
表示	☐継承されたメンバー ☐イベント発生元

FlexSheetValueFilter オブジェクトの設定の確認や変更に使用されるエディタ。

このクラスは**FlexSheetFilter** クラスによって使用されます。このクラスをユーザーコードで直接使用することはほとんどありません。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- clearEditor
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, filter: ValueFilter): ValueFilterEditor
```

ValueFilterEditor クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のセレクター（例: '#theCtrl'）。

- **filter**: **ValueFilter**
  編集する**ValueFilter** 。

継承元 **ValueFilterEditor**
戻り値 **ValueFilterEditor**

プロパティ

STATIC controlTemplate

ColumnFilterEditor コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **ValueFilterEditor**
型 **any**

filter

編集する**ValueFilter** への参照を取得します。

継承元 **ValueFilterEditor**
型 **ValueFilter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

clearEditor

```
clearEditor(): void
```

フィルタに変更を適用せずにエディタをクリアします。

継承元 **ValueFilterEditor**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

updateEditor

```
updateEditor(): void
```

現在のフィルタ設定でエディタを更新します。

戻り値 **void**

updateFilter

```
updateFilter(): void
```

現在のエディタ値を反映するようにフィルタを更新します。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

HeaderRow クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **Row**
表示 ☐継承されたメンバー ☐イベント発生元

バインドされたシートで列ヘッダ情報の表示に使用される行を表します。

コンストラクタ

- constructor

プロパティ

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- dataItem
- grid
- height
- index
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- multiLine
- pos
- renderHeight
- renderSize
- size
- visible
- visibleIndex
- wordWrap

メソッド

- onPropertyChanged

コンストラクタ

constructor

```
constructor(): HeaderRow
```

HeaderRowクラスの新しいインスタンスを初期化します。

戻り値 **HeaderRow**

プロパティ

allowDragging

ユーザーがマウスで行または列を新しい位置に移動できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

allowMerging

行または列にあるセルを結合できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

allowResizing

ユーザーがマウスで行または列をサイズ変更できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

collectionView

この行または列にバインドされた**ICollectionView** を取得します。

継承元 **RowCol**
型 **ICollectionView**

cssClass

行または列のヘッダ以外のセルをレンダリングするときに使用するCSSクラス名を取得または設定します。

継承元 **RowCol**
型 **string**

dataItem

項目がバインドされているデータコレクション内の項目を取得または設定します。

継承元 **Row**
型 **any**

grid

行または列を所有する**FlexGrid** を取得します。

継承元 **RowCol**
型 **FlexGrid**

height

行または列の高さを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

継承元 **Row**
型 **number**

- index

行または列の親コレクション内でのインデックスを取得します。

継承元 **RowCol**
型 **number**

- isContentHtml

この行または列にあるセルがプレーンテキストではなくHTMLコンテンツを含むかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isReadOnly

行または列のセルを編集できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isSelected

行または列が選択されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- isVisible

行または列が表示可能で、なおかつ折りたたまれていないかどうかを示す値を取得または設定します。

このプロパティは読み取り専用です。行または列の表示/非表示設定を変更するには、代わりに**`visible`** プロパティを使用してください。

継承元 **RowCol**
型 **boolean**

- multiLine

この行または列にあるセルの内容が改行文字(\n).でラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- pos

行または列の位置を取得します。

継承元 **RowCol**
型 **number**

renderHeight

行のレンダリング高さを取得します。

列の表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮した幅が返されます。

継承元 **Row**
型 **number**

renderSize

行または列のレンダリングサイズを取得します。 表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮したサイズが返されます。

継承元 **RowCol**
型 **number**

size

行または列のサイズを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

継承元 **RowCol**
型 **number**

visible

行または列が表示されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

visibleIndex

非表示にした要素(**`isVisible`**)を無視し、親コレクション内の行または列のインデックスを取得します。

継承元 **RowCol**
型 **number**

wordWrap

この行または列のセルの内容を使用可能な列幅に収まれるようにラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

メソッド

onPropertyChanged

```
onPropertyChanged(): void
```

オーナーリストをダーティとしてマークし、オーナーグリッドを更新します。

継承元 **RowCol**
戻り値 **void**

RowColumnChangedEventArgs クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

行または列変更イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- added
- count
- empty
- index

コンストラクタ

constructor

```
constructor(index: number, count: number, added: boolean): RowColumnChangedEventArgs
```

RowColumnChangedEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **index**: **number**
  変更された行または列の開始インデックス。
- **count**: **number**
  追加または削除された行または列の数。
- **added**: **boolean**
  行または列を追加または削除するためのイベントであることを示す値。

戻り値 **RowColumnChangedEventArgs**

プロパティ

- added

行または列を追加または削除するためのイベントであることを示す値を取得します。

型 **boolean**

- count

追加または削除された行または列の数を取得します。

型 **number**

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

index

変更された行または列の開始インデックスを取得します。

型 **number**

Sheet クラス

ファイル `wijmo.grid.sheet.js`
モジュール **wijmo.grid.sheet**
派生クラス **WjSheet**

FlexSheet コントロール内のシートを表します。

コンストラクタ

- constructor

プロパティ

- columnCount
- filterSetting
- grid
- itemsSource
- name
- rowCount
- selectionRanges
- tables
- visible

メソッド

- addTableFromArray
- dispose
- findTable
- getCellStyle
- onNameChanged
- onVisibleChanged

イベント

- nameChanged
- visibleChanged

コンストラクタ

constructor

```
constructor(owner?: FlexSheet, grid?: FlexGrid, sheetName?: string, rows?: number, cols?: number): Sheet
```

Sheet クラスの新しいインスタンスを初期化します。

パラメーター

- **owner**: **FlexSheet** OPTIONAL
  オーナーである**FlexSheet** コントロール。
- **grid**: **FlexGrid** OPTIONAL
  シートデータを格納するために使用される、関連付けられた**FlexGrid** コントロール。 指定されていない場合は、新しい**FlexGrid**コントロールが作成されます。
- **sheetName**: **string** OPTIONAL
  FlexSheet コントロール内のシートの名前。
- **rows**: **number** OPTIONAL
  シートの行数。
- **cols**: **number** OPTIONAL
  シートの列数。

戻り値 **Sheet**

プロパティ

columnCount

シート内の列数を取得または設定します。

型 **number**

filterSetting

Gets or sets the filter setting for this sheet.

型 **IFilterSetting**

grid

シートデータを格納するために使用される、関連付けられた**FlexGrid** コントロールを取得します。

型 **FlexGrid**

itemsSource

シートの**FlexGrid** インスタンスの配列または**ICollectionView** を取得または設定します。

型 **any**

name

シートの名前を取得または設定します。

型 **string**

- rowCount

シート内の行数を取得または設定します。

型 **number**

- selectionRanges

選択項目の配列を取得します。

型 **ObservableArray**

- tables

このシート内の**Table** オブジェクトのコレクションを取得します。 テーブルのコレクションを介してこのシートに **Table** を挿入または削除することができます。

型 **ObservableArray**

- visible

シートの表示/非表示設定を取得または設定します。

型 **boolean**

メソッド

addTableFromArray

```
addTableFromArray(row: number, column: number, array: any[], properties?: string[], tableName?: string, tableStyle?: TableStyle, options?: ITableOptions, shift?: boolean): Table
```

オブジェクト配列からテーブルを追加します。

パラメーター

- **row**: **number**
  テーブルの行位置。
- **column**: **number**
  テーブルの列位置。
- **array**: **any[]**
  テーブルにロードするオブジェクト配列。
- **properties**: **string[]** OPTIONAL
  これにより、配列のオブジェクトから列のサブセットのみを取得することができます。 省略すると、テーブルは配列のオブジェクトのすべてのキーをロードします。
- **tableName**: **string** OPTIONAL
  テーブルの名前。
- **tableStyle**: **TableStyle** OPTIONAL
  テーブルスタイルがテーブルに適用されます。
- **options**: **ITableOptions** OPTIONAL
  テーブルのオプション **`ITableOptions`** 。
- **shift**: **boolean** OPTIONAL
  テーブルの下のセルを移動する必要があるかどうかを示します。 指定しない場合、下にあるセルが移動されます。

戻り値 **`Table`**

dispose

```
dispose(): void
```

Dispose sheet instance.

戻り値 **void**

findTable

```
findTable(rowIndex: number, columnIndex: number): Table
```

セルの位置を介してテーブルを検索します。

パラメーター

- **rowIndex**: **number**
  指定したセルの行インデックス。
- **columnIndex**: **number**
  指定したセルの列インデックス。

戻り値 **`Table`**

getCellStyle

```
getCellStyle(rowIndex: number, columnIndex: number): ICellStyle
```

指定したセルのスタイルを取得します。

パラメーター

- **rowIndex**: **number**
  指定したセルの行インデックス。
- **columnIndex**: **number**
  指定したセルの列インデックス。

戻り値 **ICellStyle**

onNameChanged

```
onNameChanged(e: PropertyChangedEventArgs): void
```

nameChanged イベントを発生させます。

パラメーター

- **e**: **PropertyChangedEventArgs**

戻り値 **void**

onVisibleChanged

```
onVisibleChanged(e: EventArgs): void
```

visibleChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs**

戻り値 **void**

イベント

nameChanged

シート名が変更された後に発生します。

引数 **PropertyChangedEventArgs**

visibleChanged

シートの表示/非表示が変更された後に発生します。

引数 **EventArgs**

SheetCollection クラス

ファイル	wijmo.grid.sheet.js
モジュール	**wijmo.grid.sheet**
基本クラス	**ObservableArray**
表示	☐継承されたメンバー ☐イベント発生元

Sheet オブジェクトのコレクションを定義します。

コンストラクタ

- constructor

プロパティ

- isUpdating
- selectedIndex

メソッド

- beginUpdate
- clear
- deferUpdate
- endUpdate
- getValidSheetName
- hide
- implementsInterface
- indexOf
- insert
- isValidSheetName
- onCollectionChanged
- onSelectedSheetChanged
- onSheetCleared
- onSheetNameChanged
- onSheetVisibleChanged
- push
- remove
- removeAt
- selectFirst
- selectLast
- selectNext
- selectPrevious
- setAt
- show
- slice
- sort
- splice

イベント

- collectionChanged
- selectedSheetChanged
- sheetCleared
- sheetNameChanged
- sheetVisibleChanged

コンストラクタ

constructor

```
constructor(data?: any[]): ObservableArray
```

ObservableArray クラスの新しいインスタンスを初期化します。

パラメーター

- **data**: **any**[] OPTIONAL
  ObservableArray に格納する項目を含む配列。

継承元 **ObservableArray**
戻り値 **ObservableArray**

プロパティ

isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

継承元 **ObservableArray**
型

selectedIndex

現在選択されているシートのインデックスを取得または設定します。

型 **number**

メソッド

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **ObservableArray**
戻り値 **void**

clear

```
clear(): void
```

SheetCollectionをクリアします。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

継承元 **ObservableArray**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **ObservableArray**
戻り値 **void**

getValidSheetName

```
getValidSheetName(currentSheet: Sheet): string
```

シートの有効な名前を取得します。

パラメーター

- **currentSheet**: **Sheet**
  有効な名前を取得する**Sheet**。

戻り値 **string**

hide

```
hide(pos: number): boolean
```

指定した位置にあるシートを非表示にします。

パラメーター

- **pos**: **number**
  非表示にするシートの位置。

戻り値 **boolean**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **ObservableArray**
戻り値 **boolean**

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

配列内で項目を検索します。

パラメーター

- **searchElement**: **any**
  配列内で検索する要素。

- **fromIndex**: **number** OPTIONAL
  検索を開始するインデックス。

継承元 **ObservableArray**
戻り値 **number**

insert

```
insert(index: number, item: any): void
```

配列内の指定した位置に項目を挿入します。 基本クラスである**ObservableArray** のinsertメソッドをオーバーライドします。

パラメーター

- **index**: **number**
  項目を追加する位置。

- **item**: **any**
  配列に追加する項目。

戻り値 **void**

isValidSheetName

```
isValidSheetName(sheet: Sheet): boolean
```

シート名が有効かどうかをチェックします。

パラメーター

- **sheet**: **Sheet**
  名前をチェックする**Sheet**。

戻り値 **boolean**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

collectionChanged イベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  変更の記述が含まれます。

継承元 **ObservableArray**
戻り値 **void**

onSelectedSheetChanged

```
onSelectedSheetChanged(e: PropertyChangedEventArgs): void
```

currentChangedイベントを発生させます。

パラメーター

- **e**: **PropertyChangedEventArgs**
  イベントデータを含む**PropertyChangedEventArgs**。

戻り値 **void**

onSheetCleared

```
onSheetCleared(): void
```

sheetClearedイベントを発生させます。

戻り値 **void**

onSheetNameChanged

```
onSheetNameChanged(e: NotifyCollectionChangedEventArgs): void
```

sheetNameChangedイベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs**

戻り値 **void**

onSheetVisibleChanged

```
onSheetVisibleChanged(e: NotifyCollectionChangedEventArgs): void
```

sheetVisibleChangedイベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs**

戻り値 **void**

push

```
push(...item: any[]): number
```

配列の末尾に1つ以上の項目を追加します。 基本クラス**ObservableArray** のpushメソッドをオーバーライドします。

パラメーター

- **...item**: **any[]**
  配列に追加する1つ以上の項目。

戻り値 **number**

remove

```
remove(item: any): boolean
```

配列から項目を削除します。

パラメーター

- **item**: **any**
  削除する項目。

継承元 **ObservableArray**
戻り値 **boolean**

removeAt

```
removeAt(index: number): void
```

配列内の指定した位置にある項目を削除します。 基本クラスである**ObservableArray** のremoveAtメソッドをオーバーライドします。

パラメーター

- **index**: **number**
  削除する項目の位置。

戻り値 **void**

selectFirst

```
selectFirst(): boolean
```

FlexSheet コントロールで最初のシートを選択します。

戻り値 **boolean**

selectLast

```
selectLast(): boolean
```

オーナーである**FlexSheet** コントロールで最後のシートを選択します。

戻り値 **boolean**

selectNext

```
selectNext(): boolean
```

オーナーである**FlexSheet** コントロールで次のシートを選択します。

戻り値 **boolean**

selectPrevious

```
selectPrevious(): boolean
```

オーナーである**FlexSheet** コントロールで前のシートを選択します。

戻り値 **boolean**

setAt

```
setAt(index: number, item: any): void
```

配列内の指定した位置にある項目を設定します。

パラメーター

- **index**: **number**
  項目を設定する位置。

- **item**: **any**
  配列に設定する項目。

継承元 **ObservableArray**
戻り値 **void**

show

```
show(pos: number): boolean
```

指定した位置にある**Sheet** の非表示を解除して選択します。

パラメーター

- **pos**: **number**
  表示するシートの位置。

戻り値 **boolean**

slice

```
slice(begin?: number, end?: number): any[]
```

配列の一部のシャローコピーを作成します。

パラメーター

- **begin**: **number** OPTIONAL
  コピーを開始する位置。

- **end**: **number** OPTIONAL
  コピーを終了する位置。

継承元 **ObservableArray**
戻り値 **any[]**

sort

```
sort(compareFn?: Function): this
```

配列の要素を適切な位置にソートします。

パラメーター

- **compareFn**: **Function** OPTIONAL
  ソート順序を定義する関数。 この関数は2つの引数を受け取り、-1（最初の引数の方が2番目の引数より小さい場合）、+1（大きい場合）、0（等しい場合）のいずれかの値を返す必要があります。 これを省略した場合は、各要素の文字列変換に従って辞書順にソートされます。

継承元 **ObservableArray**
戻り値 **this**

splice

```
splice(index: number, count: number, item?: any): any[]
```

配列からの項目の削除と配列への項目の追加の一方または両方を行います。 基本クラス**ObservableArray** のspliceメソッドをオーバーライドします。

パラメーター

- **index**: **number**
  項目を追加または削除する位置。

- **count**: **number**
  配列から削除する項目の数。

- **item**: **any** OPTIONAL
  配列に追加する項目。

戻り値 **any[]**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 **ObservableArray**
引数 **NotifyCollectionChangedEventArgs**

selectedSheetChanged

selectedIndexプロパティが変更されたときに発生します。

引数 **PropertyChangedEventArgs**

sheetCleared

SheetCollection がクリアされたときに発生します。

引数 **EventArgs**

sheetNameChanged

コレクション内のシートの名前が変更された後に発生します。

引数 **NotifyCollectionChangedEventArgs**

sheetVisibleChanged

コレクション内のシートの表示/非表示が変更された後に発生します。

引数 **NotifyCollectionChangedEventArgs**

SortManager クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

FlexSheet で選択されている**Sheet** のソートを管理します。

コンストラクタ

- constructor

プロパティ

- sortDescriptions

メソッド

- addSortLevel
- cancelSort
- checkSortItemExists
- clearSort
- commitSort
- copySortLevel
- deleteSortLevel
- editSortLevel
- moveSortLevel

コンストラクタ

constructor

```
constructor(owner: FlexSheet): SortManager
```

SortManager クラスの新しいインスタンスを初期化します。

パラメーター

- **owner**: **FlexSheet**
  The **FlexSheet** control that owns this [[SDLENTITYREF[lt]]]b[[SDLENTITYREF[gt]]]SortManager[[SDLENTITYREF[lt]]]/b[[SDLENTITYREF[gt]]].

戻り値 **SortManager**

プロパティ

- sortDescriptions

ColumnSortDescription オブジェクトによって表されたソート記述のコレクションを取得または設定します。

型 **CollectionView**

メソッド

addSortLevel

```
addSortLevel(columnIndex?: number, ascending?: boolean): void
```

ソート記述に空白のソートレベルを追加します。

パラメーター

- **columnIndex**: **number** OPTIONAL
  FlexSheetコントロールの列のインデックス。

- **ascending**: **boolean** OPTIONAL
  ソートレベルのソート順序。

戻り値 **void**

cancelSort

```
cancelSort(): void
```

FlexSheetコントロールに対して現在のソート記述をキャンセルします。

戻り値 **void**

checkSortItemExists

```
checkSortItemExists(columnIndex): number
```

特定の列のソート項目が存在するかどうかをチェックします。

パラメーター

- **columnIndex**:
  FlexSheetコントロールの列のインデックス。

戻り値 **number**

clearSort

```
clearSort(): void
```

ソート記述をクリアします。

戻り値 **void**

commitSort

```
commitSort(undoable?: boolean): void
```

現在のソート記述をFlexSheetコントロールにコミットします。

パラメーター

- **undoable**: **boolean** OPTIONAL
  ソートのコミット操作を元に戻せるかどうかを示すブール値。

戻り値 **void**

copySortLevel

```
copySortLevel(): void
```

ソート記述に現在のソートレベルのコピーを追加します。

戻り値 **void**

deleteSortLevel

```
deleteSortLevel(columnIndex?: number): void
```

ソート記述から現在のソートレベルを削除します。

パラメーター

- **columnIndex**: **number** OPTIONAL
  FlexSheetコントロールの列のインデックス。

戻り値 **void**

editSortLevel

```
editSortLevel(columnIndex?: number, ascending?: boolean): void
```

現在のソートレベルを更新します。

パラメーター

- **columnIndex**: **number** OPTIONAL
  ソートレベルの列インデックス。

- **ascending**: **boolean** OPTIONAL
  ソートレベルのソート順序。

戻り値 **void**

moveSortLevel

```
moveSortLevel(offset: number): void
```

現在のソートレベルを新しい位置に移動します。

パラメーター

- **offset**: **number**
  現在のレベルを移動するオフセット。

戻り値 **void**

Table クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

FlexSheet コントロール内のテーブルを表します。

コンストラクタ

- constructor

プロパティ

- alterFirstColumn
- alterLastColumn
- name
- sheet
- showBandedColumns
- showBandedRows
- showHeaderRow
- showTotalRow
- style

メソッド

- deleteRows
- getColumns
- getRange
- insertRows

コンストラクタ

constructor

```
constructor(name: string, range: CellRange, style?: TableStyle, columns?: TableColumn[], options?: ITableOptions): Table
```

Table クラスの新しいインスタンスを初期化します。

パラメーター

- **name**: **string**
  テーブルの名前。
- **range**: **CellRange**
  テーブルの範囲。
- **style**: **TableStyle** OPTIONAL
  テーブルで使用するテーブルスタイル。 スタイルが省略されている場合、デフォルトのスタイルとして組み込みの'TableStyleMedium9'テーブルスタイルが適用されます。
- **columns**: **TableColumn**[] OPTIONAL
  テーブルの列。
- **options**: **ITableOptions** OPTIONAL
  テーブルのオプション**ITableOptions** 。

戻り値 **Table**

プロパティ

- alterFirstColumn

最初のテーブルの列にスタイルを適用するかどうかを判定する値を取得または設定します。

型 **boolean**

- alterLastColumn

最後のテーブルの列にスタイルを適用するかどうかを判定する値を取得または設定します。

型 **boolean**

- name

テーブル名を取得または設定します。

テーブル名は、プログラムによってテーブルを参照するために使用されます。

型 **string**

- sheet

このテーブルが属する **`Sheet`** を取得します。

型 **`Sheet`**

- showBandedColumns

縞模様 (列)の書式設定が適用されるかどうかを示します。

型 **boolean**

- showBandedRows

縞模様(行)の書式設定が適用されるかどうかを判定する値を取得または設定します。

型 **boolean**

- showHeaderRow

テーブルに見出し行を含めるかどうかを示します。

型 **boolean**

- showTotalRow

テーブルに集計行を含めるかどうかを示します。

型 **boolean**

style

このテーブルに関連付けられた**TableStyle** を取得または設定します。

型 **TableStyle**

メソッド

deleteRows

```
deleteRows(index: number, count?: number, shift?: boolean): void
```

テーブルの行を削除します。

パラメーター

- **index**: **number**
  削除するテーブル行の開始インデックス。
- **count**: **number** OPTIONAL
  削除する行数。 これを指定しない場合、1行が削除されます。
- **shift**: **boolean** OPTIONAL
  テーブルの下のセルを移動する必要があるかどうかを示します。 指定しない場合、下にあるセルが移動されます。

戻り値 **void**

getColumns

```
getColumns(): TableColumn[]
```

テーブルの列を取得します。

戻り値 **TableColumn**[]

getRange

```
getRange(section?: TableSection, column?: any): CellRange
```

テーブルが占める関連シートの特定のセクションと列の範囲を取得します。

パラメーター

- **section**: **TableSection** OPTIONAL
  テーブルのセクション。 セクションを省略した場合、 テーブル全体の範囲を取得します。
- **column**: **any** OPTIONAL
  テーブルの列。 列は、 **TableColumn** のインスタンス、列名、または列インデックスにすることができます。 列を省略した場合、 テーブル内のすべての列の範囲を取得します。 セクションがnullの場合、特定の列の参照に見出しと集計行（表示されている場合）が含まれます。

戻り値 **CellRange**

insertRows

```
insertRows(index: number, count?: number, shift?: boolean): boolean
```

テーブルに行を挿入します。

パラメーター

- **index**: **number**
  新しい行をテーブルに追加する位置。

- **count**: **number** OPTIONAL
  追加する行数。これを指定しない場合、1行が追加されます。

- **shift**: **boolean** OPTIONAL
  テーブルの下のセルを移動する必要があるかどうかを示します。 指定しない場合、下にあるセルが移動されます。

戻り値 **boolean**

TableColumn クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

Table 内の列を表します。

コンストラクタ

- constructor

プロパティ

- name
- showFilterButton
- table
- totalRowFunction
- totalRowLabel

コンストラクタ

constructor

```
constructor(name: string, totalRowLabel?: string, totalRowFunction?: string, showFilterButton?: boolean): TableColumn
```

TableColumn クラスの新しいインスタンスを初期化します。

パラメーター

- **name**: **string**
  テーブル列の名前。
- **totalRowLabel**: **string** OPTIONAL
  列に該当する合計行のセルに表示する文字列。
- **totalRowFunction**: **string** OPTIONAL
  列に該当する合計行のセルに表示する関数。
- **showFilterButton**: **boolean** OPTIONAL
  テーブル列に対してフィルタのボタンを表示するかどうかを示します。 showFilterButtonのデフォルト値はtrueです。

戻り値 **TableColumn**

プロパティ

- name

テーブル列の名前を取得または設定します。列名は、関数によって参考されます。

型 **string**

- showFilterButton

テーブル列に対してフィルタのボタンを表示するかどうかを示します。

FlexSheetはテーブルのフィルタをまだサポートしていないため、このプロパティはインポート/エクスポート操作にのみ使用されます。

型 **boolean**

- table

テーブル列が属するTableを取得します。

型 `Table`

- totalRowFunction

列に該当する合計行のセルに表示する関数。

型 **string**

- totalRowLabel

列に該当する合計行のセルに表示する文字列。

型 **string**

TableStyle クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

Table に対するテーブルスタイルを表します。

コンストラクタ

- constructor

プロパティ

- firstBandedColumnStyle
- firstBandedRowStyle
- firstColumnStyle
- firstHeaderCellStyle
- firstTotalCellStyle
- headerRowStyle
- isBuiltIn
- lastColumnStyle
- lastHeaderCellStyle
- lastTotalCellStyle
- name
- secondBandedColumnStyle
- secondBandedRowStyle
- totalRowStyle
- wholeTableStyle

コンストラクタ

constructor

```
constructor(name: string, isBuiltIn?: boolean): TableStyle
```

TableStyle クラスの新しいインスタンスを初期化します。

パラメーター

- **name**: **string**
  テーブルスタイルの名前。

- **isBuiltIn**: **boolean** OPTIONAL
  テーブルのスタイルが組み込みスタイルかどうかを示します。

戻り値 **TableStyle**

プロパティ

- firstBandedColumnStyle

最初の横模様の列のスタイルを取得または設定します。

型 **IBandedTableSectionStyle**

- firstBandedRowStyle

最初の横模様の行のスタイルを取得または設定します。

型 **IBandedTableSectionStyle**

- firstColumnStyle

最初列のスタイルを取得または設定します。

型 **ITableSectionStyle**

- firstHeaderCellStyle

見出し行の最初セルのスタイルを取得または設定します。

型 **ITableSectionStyle**

- firstTotalCellStyle

集計行の最初セルのスタイルを取得または設定します。

型 **ITableSectionStyle**

- headerRowStyle

見出し行のスタイルを取得または設定します。

型 **ITableSectionStyle**

- isBuiltIn

テーブルのスタイルが組み込みスタイルかどうかを示します。

型 **boolean**

- lastColumnStyle

最終列のスタイルを取得または設定します。

型 **ITableSectionStyle**

- lastHeaderCellStyle

見出し行のっ最終セルのスタイルを取得または設定します。

型 **ITableSectionStyle**

- lastTotalCellStyle

集計行の最終セルのスタイルを取得または設定します。

型 **ITableSectionStyle**

- name

テーブルスタイルの名前を取得または設定します。

型 **string**

- secondBandedColumnStyle

２番目の横模様の列のスタイルを取得または設定します。

型 **IBandedTableSectionStyle**

- secondBandedRowStyle

2番目の横模様の行のスタイルを取得または設定します。

型 **IBandedTableSectionStyle**

- totalRowStyle

集計行のスタイルを取得または設定します。

型 **ITableSectionStyle**

- wholeTableStyle

テーブル全体のスタイルを取得または設定します。

型 **ITableSectionStyle**

UndoStack クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

FlexSheet の元に戻す/やり直し操作を制御します。

コンストラクタ

- constructor

プロパティ

- canRedo
- canUndo
- stackSize

メソッド

- clear
- onUndoStackChanged
- redo
- undo

イベント

- undoStackChanged

コンストラクタ

constructor

```
constructor(owner: FlexSheet): UndoStack
```

UndoStack クラスの新しいインスタンスを初期化します。

パラメーター

- **owner**: **FlexSheet**
  この**UndoStack** によって管理する**FlexSheet** コントロール。

戻り値 **UndoStack**

プロパティ

canRedo

やり直し操作を実行できるかどうかをチェックします。

型 **boolean**

canUndo

元に戻す操作を実行できるかどうかをチェックします。

型 **boolean**

stackSize

元に戻すスタックのサイズを取得または設定します。

型 **number**

メソッド

clear

```
clear(): void
```

アンドゥスタックをクリアします。

戻り値 **void**

onUndoStackChanged

```
onUndoStackChanged(): void
```

undoStackChanged イベントを発生させます。

戻り値 **void**

redo

```
redo(): void
```

最後に元に戻された操作をやり直します。

戻り値 **void**

undo

```
undo(): void
```

最後の操作を元に戻します。

戻り値 **void**

イベント

undoStackChanged

アンドゥスタックが変更された後に発生します。

引数 **EventArgs**

UnknownFunctionEventArgs クラス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

不明な関数イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- empty
- funcName
- params
- value

コンストラクタ

constructor

```
constructor(funcName: string, params: any[]): UnknownFunctionEventArgs
```

UnknownFunctionEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **funcName**: **string**
  不明な関数の名前。
- **params**: **any**[]
  不明な関数のパラメータの値リスト。

戻り値 **UnknownFunctionEventArgs**

プロパティ

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

- funcName

不明な関数の名前を取得します。

型 **string**

- params

不明な関数のパラメータの値リストを取得します。

型 **any**[]

value

不明な関数の結果を取得または設定します。

型 **string**

IBandedTableSectionStyle インターフェイス

ファイル `wijmo.grid.sheet.js`
モジュール **wijmo.grid.sheet**
インターフェイス **ITableSectionStyle**

テーブルのスタイリングをストライプに設定するプロパティを定義します。

プロパティ

- size

プロパティ

- size

単一のストライプバンド内の行数または列数。

型 **number**

ICellStyle インターフェイス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

セルのスタイルを設定するプロパティを定義します。

プロパティ

- backgroundColor
- borderBottomColor
- borderBottomStyle
- borderBottomWidth
- borderLeftColor
- borderLeftStyle
- borderLeftWidth
- borderRightColor
- borderRightStyle
- borderRightWidth
- borderTopColor
- borderTopStyle
- borderTopWidth
- className
- color
- fontFamily
- fontSize
- fontStyle
- fontWeight
- format
- textAlign
- textDecoration
- verticalAlign
- whiteSpace

プロパティ

backgroundColor

背景色。

型 **string**

borderBottomColor

下側の枠線の色。

型 **string**

- borderBottomStyle

下側の枠線のスタイル。

型 **string**

- borderBottomWidth

下側の枠線の幅。

型 **string**

- borderLeftColor

左側の枠線の色。

型 **string**

- borderLeftStyle

左側の枠線のスタイル。

型 **string**

- borderLeftWidth

左側の枠線の幅。

型 **string**

- borderRightColor

右側の枠線の色。

型 **string**

- borderRightStyle

右側の枠線のスタイル。

型 **string**

- borderRightWidth

右側の枠線の幅。

型 **string**

- borderTopColor

上側の枠線の色。

型 **string**

- borderTopStyle

上側の枠線のスタイル。

型 **string**

- borderTopWidth

上側の枠線の幅。

型 **string**

- className

セルに追加するCSSクラス名。

型 **string**

- color

フォントの色。

型 **string**

- fontFamily

フォントファミリー。

型 **string**

- fontSize

フォントサイズ。

型 **string**

- fontStyle

フォントスタイル。

型 **string**

- fontWeight

フォントウェイト。

型 **string**

- format

セルの値を書式設定する書式文字列。

型 **string**

- textAlign

テキストの配置。

型 **string**

- textDecoration

テキスト装飾。

型 **string**

- verticalAlign

垂直方向の配置。

型 **string**

- whiteSpace

要素内の空白を処理する方法を記述します。

型 **string**

IColumnFilterSetting インターフェイス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

The setting for column filter.

プロパティ

- column
- conditionFilterSetting
- dataMap
- filterType
- valueFilterSetting

プロパティ

column

フィルタリングされる列。It could be the **Column** instance, name of the **Column** or index in the column collection.

型 **any**

conditionFilterSetting

The condition filter setting in this column filter setting.

型 **IConditionFilterSetting**

dataMap

未加工の値をこのフィルタを編集する際に表示される表示値に変換するために使用される**DataMap** 。

型 **DataMap**

filterType

このフィルタから提供されるフィルタ処理のタイプ。

型 **FilterType**

valueFilterSetting

The value filter setting in this column filter setting.

型 **IValueFiterSetting**

IConditionFilterSetting インターフェイス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

The condition filter setting.

プロパティ

- dataMap

プロパティ

dataMap

未加工の値をこのフィルタを編集する際に表示される表示値に変換するために使用される**DataMap**。

型 **DataMap**

IFilterSetting インターフェイス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

Defines the filter setting of sheet.

プロパティ

- columnFilterSettings
- filterColumns

プロパティ

- columnFilterSettings

The filter setting for the columns of the sheet.

型 **IColumnFilterSetting**[]

- filterColumns

フィルタを持つ列の名前またはバインディングを含む配列。

型 **string**[]

IFlexSheetXlsxOptions インターフェイス

ファイル `wijmo.grid.sheet.js`
モジュール **wijmo.grid.sheet**

FlexSheet Xlsx export options

プロパティ

- includeFormulaValues

プロパティ

includeFormulaValues

Indicates whether export the calculated value for formula cells.

型 **boolean**

IFormatState インターフェイス

ファイル wijmo.grid.sheet.js
モジュール **wijmo.grid.sheet**

Defines the format states for the cells.

プロパティ

- isBold
- isItalic
- isMergedCell
- isUnderline
- textAlign

プロパティ

isBold

Indicates whether the bold style is applied.

型 **boolean**

isItalic

Indicates whether the italic style is applied.

型 **boolean**

isMergedCell

Indicate whether the current selection is a merged cell.

型 **boolean**

isUnderline

Indicates whether the underlined style is applied.

型 **boolean**

textAlign

Gets the applied text alignment.

型 **string**

ITableOptions インターフェイス

ファイル `wijmo.grid.sheet.js`
モジュール **wijmo.grid.sheet**

Defines the table options for creating table.

プロパティ

- alterFirstColumn
- alterLastColumn
- showBandedColumns
- showBandedRows
- showHeaderRow
- showTotalRow

プロパティ

alterFirstColumn

Indicating whether the first column in the table should have the style applied.

型 **boolean**

alterLastColumn

Indicating whether the last column in the table should have the style applied.

型 **boolean**

showBandedColumns

Indicating whether banded column formatting is applied.

型 **boolean**

showBandedRows

Indicating whether banded row formatting is applied.

型 **boolean**

showHeaderRow

Indicates whether show the header row for the table.

型 **boolean**

showTotalRow

Indicates whether show the total row for the table.

型 **boolean**

ITableSectionStyle インターフェイス

ファイル `wijmo.grid.sheet.js`
モジュール **wijmo.grid.sheet**
インターフェイス **ICellStyle**

Defines the table styling properties.

プロパティ

- borderHorizontalColor
- borderHorizontalStyle
- borderHorizontalWidth
- borderVerticalColor
- borderVerticalStyle
- borderVerticalWidth

プロパティ

borderHorizontalColor

Color of the Horizontal border.

型 **any**

borderHorizontalStyle

Style of the Horizontal border.

型 **string**

borderHorizontalWidth

Width of the Horizontal border.

型 **string**

borderVerticalColor

Color of the Vertical border.

型 **any**

borderVerticalStyle

Style of the Vertical border.

型 **string**

borderVerticalWidth

Width of the Vertical border.

型 **string**

IValueFiterSetting インターフェイス

ファイル `wijmo.grid.sheet.js`
モジュール **wijmo.grid.sheet**

The value filter setting.

プロパティ

- dataMap
- maxValues
- sortValues
- uniqueValues

プロパティ

dataMap

The **DataMap** used to convert raw values into display values shown when editing this filter.

型 **DataMap**

maxValues

The maximum number of elements on the list of display values.

型 **number**

sortValues

A value that determines whether the values should be sorted

型 **boolean**

uniqueValues

An array containing the unique values to be displayed on the list.

型 **any**[]

TableSection 列挙体

ファイル `wijmo.grid.sheet.js`
モジュール **wijmo.grid.sheet**

テーブルのセクションを定義する定数を指定します。

メンバー

名前	値	説明
All	0	ヘッダ、データ、フッタを含むテーブル全体。
Data	1	データ行。
Header	2	ヘッダ行。
Footer	3	フッタ行。

wijmo.chart.finance モジュール

ファイル `wijmo.chart.finance.js`
モジュール **wijmo.chart.finance**

FinancialChart コントロールとそのコントロールに関連付けられたクラスを定義します。

クラス

- FinancialChart
- FinancialSeries

列挙体

- DataFields
- FinancialChartType
- PointAndFigureScaling
- RangeMode

FinancialChart クラス

ファイル wijmo.chart.finance.js
モジュール wijmo.chart.finance
基本クラス FlexChartCore
派生クラス WjFinancialChart
表示 ☐継承されたメンバー ☐イベント発生元

株価チャートコントロール。

コンストラクタ

- constructor

プロパティ

- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- interpolateNulls
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- options
- palette
- plotAreas
- plotMargin
- rightToLeft
- selection
- selectionMode
- series
- symbolSize
- tooltip

メソッド

- addEventListener
- applyTemplate
- beginUpdate

- containsFocus
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FinancialChart
```

FlexChart クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  このコントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **FinancialChart**

プロパティ

axes

Axis オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

axisX

メインのX軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

axisY

メインのY軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

binding

Yの値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

bindingX

Xデータ値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

chartType

作成する株価チャートのタイプを取得または設定します。

型 **FinancialChartType**

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

- dataLabel

ポイントのデータラベルを取得または設定します。

継承元 **FlexChartCore**
型 **DataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 `FlexChartCore`
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

legendToggle

凡例項目をクリックしたときにチャート上の系列の表示/非表示を切り替えるかどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 `FlexChartCore`
型 **boolean**

options

さまざまなチャートオプションを取得または設定します。

以下のオプションがサポートされます。

kagi.fields：カギ足チャートで使用される**`DataFields`** を指定します。デフォルト値はDataFields.Closeです。

kagi.rangeMode：カギ足チャートで使用される**`RangeMode`** を指定します。デフォルト値はRangeMode.Fixedです。

kagi.reversalAmount：カギ足チャートの反転量を指定します。デフォルト値は14です。

```
chart.options = {
  kagi: {
     fields: wijmo.chart.finance.DataFields.Close,
     rangeMode: wijmo.chart.finance.RangeMode.Fixed,
     reversalAmount: 14
  }
}
```

lineBreak.newLineBreaks:：ラインブレイクチャートで、何個のボックスを 比較してから新しいボックスを描画するかを 取得または設定します。デフォルト値は3です。

```
chart.options = {
  lineBreak: { newLineBreaks: 3 }
}
```

renko.fields：練行足チャートで使用される**`DataFields`** を指定します。デフォルト値はDataFields.Closeです。

renko.rangeMode：練行足チャートで使用される**`RangeMode`** を指定します。デフォルト値はRangeMode.Fixedです。

renko.boxSize：練行足チャートのボックスサイズを指定します。デフォルト値は14です。

```
chart.options = {
  renko: {
     fields: wijmo.chart.finance.DataFields.Close,
     rangeMode: wijmo.chart.finance.RangeMode.Fixed,
     boxSize: 14
  }
}
```

型 **any**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string**[]

plotAreas

PlotArea オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **PlotAreaCollection**

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selection

選択されているチャート系列を取得または設定します。

継承元 **FlexChartCore**
型 **SeriesBase**

selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

series

Series オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

symbolSize

この**FlexChart** のすべてのSeriesオブジェクトに使用されるシンボルのサイズを取得または設定します。

このプロパティは、各**Series** オブジェクトのsymbolSizeプロパティによってオーバーライドできます。

The default value for this property is **10** pixels.

継承元 **FlexChartCore**
型 **number**

tooltip

チャートの**Tooltip** オブジェクトを取得します。

ツールチップの内容は、次のパラメータを含むテンプレートを使用して生成されます。

- **propertyName**： このポイントによって表されるデータオブジェクトの任意のプロパティ。
- **seriesName**： データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**： データポイントのインデックス。
- **value**： データポイントの**値**（**FlexChart** の場合はy値、**FlexPie** の場合は項目値）。
- **x**： データポイントの**x**値（FlexChartのみ）。
- **y**： データポイントの**y**値（FlexChartのみ）。
- **name**： データポイントの**名前**（**FlexChart** の場合はx値、**FlexPie** の場合は凡例エントリ）。

テンプレートを変更するには、ツールチップのコンテンツプロパティに新しい値を割り当てます。 次に例を示します。

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

チャートのツールチップを無効にできます。それには、テンプレートを空の文字列に設定します。

tooltip プロパティを使用して、次のように、**showDelay** や**hideDelay** などの ツールチップパラメータをカスタマイズすることもできます。

```
chart.tooltip.showDelay = 1000;
```

詳細とオプションについては、**ChartTooltip** プロパティを参照してください。

継承元　　**FlexChartCore**
型　　**ChartTooltip**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Point をデータ座標からコントロール座標に変換します。

パラメーター

- **pt**: **any**
  データ座標の**Point** 、またはデータ座標のポイントのX座標。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

seriesVisibilityChanged イベントを発生させます。

パラメーター

- **e**: **SeriesEventArgs**
  イベントデータを含む**SeriesEventArgs** オブジェクト。

継承元 **FlexChartCore**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

pointToData

```
pointToData(pt: any, y?: number): Point
```

Point をコントロール座標からチャートデータ座標に変換します。

パラメーター

- **pt**: **any**
  変換するポイント（コントロール座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**`ImageFormat`** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **`FlexChartBase`**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **`FlexChartBase`**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

seriesVisibilityChanged

系列の表示/非表示が変更されたときに発生します。たとえば、legendToggleプロパティをtrueに設定したり、ユーザーが凡例をクリックしたときです。

継承元 **FlexChartCore**
引数 **SeriesEventArgs**

FinancialSeries クラス

ファイル	wijmo.chart.finance.js
モジュール	**wijmo.chart.finance**
基本クラス	**SeriesBase**
派生クラス	**WjFinancialChartSeries**
表示	☐継承されたメンバー ☐イベント発生元

チャートに表示するデータポイントの系列を表します。

Series クラスは、基本的なチャートタイプのすべてをサポートします。**FlexChart** 系列コレクションに追加する**Series** オブジェクトごとに異なるチャートタイプを定義できます。**Series** オブジェクトに異なるチャートタイプを設定すると、チャートに設定された**chartType** プロパティ（系列コレクション内のすべての**Series** オブジェクトのデフォルト）がオーバーライドされます。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): SeriesBase
```

SeriesBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **SeriesBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

chartType

特定の系列のチャートタイプを取得または設定します。これはチャート全体に設定されたチャートタイプをオーバーライドします。ColumnVolume、EquiVolume、CandleVolume、ArmsCandleVolumeの各チャートタイプはサポートされていない点に注意してください。これらのチャートタイプは**FinancialChart** に対して設定する必要があります

型 **FinancialChartType**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

DataFields 列挙体

ファイル `wijmo.chart.finance.js`
モジュール **wijmo.chart.finance**

計算に使用されるフィールドを指定します。練行足およびカギ足チャートタイプに適用されます。

メンバー

名前	値	説明
Close	0	計算に終値が使用されます。
High	1	計算に高値が使用されます。
Low	2	計算に安値が使用されます。
Open	3	計算に始値が使用されます。
HighLow	4	計算にHigh-Lowメソッドが使用されます。 現在、練行足チャートタイプはDataFields.HighLowをサポートしていません。
HL2	5	計算に高値および安値の平均が使用されます。
HLC3	6	計算に高値、安値、終値の平均が使用されます。
HLOC4	7	計算に高値、安値、始値、終値の平均が使用されます。

FinancialChartType 列挙体

ファイル `wijmo.chart.finance.js`
モジュール **wijmo.chart.finance**

株価チャートのタイプを指定します。

メンバー

名前	値	説明
Column	0	縦棒を表示して、カテゴリ間で項目の値を比較できるようにします。
Scatter	1	X座標とY座標を使用して、データのパターンを表示します。
Line	2	一定期間のトレンドまたはカテゴリ間のトレンドを表示します。
LineSymbols	3	各データポイントにシンボルを使用する折れ線グラフを表示します。
Area	4	線の下の領域が色で塗りつぶされた折れ線グラフを表示します。
Candlestick	5	高値、安値、始値、終値を持つ項目を表示します。 ヒゲ線のサイズは高値と安値で決定され、 棒のサイズは、始値と終値で決定されます。 胴体の色は、終値が始値より高いか低いかによって異なる色になります。 このチャートタイプのデータは、**FlexChart** プロパティまたは**Series binding**プロパティを使用して、 "yProperty, bubbleSizeProperty"形式のカンマ区切り値で定義できます。
HighLowOpenClose	6	ローソク足チャートと同じ情報を表示しますが、 始値が左向きの線、終値が右向きの線 で表されます このチャートタイプのデータは、 **FlexChart** プロパティまたは**Series binding**プロパティを使用して、 "yProperty, bubbleSizeProperty"形式のカンマ区切り値で定義できます。
HeikinAshi	7	ローソク足チャートから派生したチャートで、ノイズを除去するために現在の期間と前の期間の情報を使用します。 これらのチャートを他の系列オブジェクトと組み合わせることはできません。 このチャートタイプのデータは、 **FlexChart** プロパティまたは**Series binding**プロパティを使用して、 "yProperty, bubbleSizeProperty"形式のカンマ区切り値で定義できます。
LineBreak	8	価格変動のみに焦点を合わせることによってノイズを除去します。 これらのチャートを他の系列オブジェクトと組み合わせることはできません。 このチャートタイプのデータは、 **FlexChart** プロパティまたは**Series binding**プロパティを使用して、 "yProperty, bubbleSizeProperty"形式のカンマ区切り値で定義できます。
Renko	9	時間を無視し、相場の動きに焦点を合わせます。 これらのチャートを他の系列オブジェクトと組み合わせることはできません。 このチャートタイプのデータは、 **FlexChart** プロパティまたは**Series binding**プロパティを使用して、 "yProperty, bubbleSizeProperty"形式のカンマ区切り値で定義できます。
Kagi	10	時間を無視し、相場の動きに焦点を合わせます。 これらのチャートを他の系列オブジェクトと組み合わせることはできません。 このチャートタイプのデータは、 **FlexChart** プロパティまたは**Series binding**プロパティを使用して、 "yProperty, bubbleSizeProperty"形式のカンマ区切り値で定義できます。
ColumnVolume	11	標準の縦棒グラフに似ていますが、それぞれの棒の幅がVolume値によって決定される点が異なります。 このチャートタイプのデータは、**FinancialChart** または**FinancialSeries** の**binding**プロパティを使用して、"yProperty, volumeProperty"の形式のカンマ区切り値として定義できます。 このチャートタイプは**FinancialChart** レベルでのみ使用できます。**FinancialSeries** オブジェクトに適用しないでください。 現時点では、**FinancialChart** あたり1セットのボリュームデータのみがサポートされています。
EquiVolume	12	ローソク足チャートに似ていますが、高値と安値のみを示します。 また、それぞれの棒の幅がVolume値によって決定されます。 このチャートタイプのデータは、**FinancialChart** または**FinancialSeries** の**binding**プロパティを使用して、"highProperty, lowProperty, openProperty, closeProperty, volumeProperty"の形式のカンマ区切り値として定義できます。 このチャートタイプは**FinancialChart** レベルでのみ使用できます。**FinancialSeries** オブジェクトに適用しないでください。 現時点では、**FinancialChart** あたり1セットのボリュームデータのみがサポートされています。
CandleVolume	13	標準のローソク足チャートに似ていますが、それぞれの棒の幅がVolume値によって決定される点が異なります。 このチャートタイプのデータは、**FinancialChart** または**FinancialSeries** の**binding**プロパティを使用して、"highProperty, lowProperty, openProperty, closeProperty, volumeProperty"の形式のカンマ区切り値として定義できます。 このチャートタイプは**FinancialChart** レベルでのみ使用できます。**FinancialSeries** オブジェクトに適用しないでください。 現時点では、**FinancialChart** あたり1セットのボリュームデータのみがサポートされています。
ArmsCandleVolume	14	Richard Arms氏によって作成されたこのチャートは、EquiVolumeチャートタイプとCandleVolumeチャートタイプを組み合わせたものです。 このチャートタイプのデータは、**FinancialChart** または**FinancialSeries** の**binding**プロパティを使用して、"highProperty, lowProperty, openProperty, closeProperty, volumeProperty"の形式のカンマ区切り値として定義できます。 このチャートタイプは**FinancialChart** レベルでのみ使用できます。**FinancialSeries** オブジェクトに適用しないでください。 現時点では、**FinancialChart** あたり1セットのボリュームデータのみがサポートされています。
PointAndFigure	15	ポイントアンドフィギュア株価チャート。このチャートタイプのデータは、**FinancialChart** または**FinancialSeries** の**binding**プロパティを使用して、"highProperty, lowProperty, closeProperty"の形式のカンマ区切り値として定義できます。 このチャートタイプは**FinancialChart** レベルでのみ使用できます。**FinancialSeries** オブジェクトに適用できません。

PointAndFigureScaling 列挙体

ファイル `wijmo.chart.finance.js`
モジュール **wijmo.chart.finance**

ポイントアンドフィギュアチャートのスケーリングモードを指定します。

メンバー

名前	値	説明
Traditional	0	標準スケーリング。 ボックスサイズは、株価範囲に基づいて自動的に計算されます。
Fixed	1	固定スケーリング。 ボックスサイズはboxSizeプロパティで定義されます。
Dynamic	2	動的(ATR)スケーリング。 ボックスサイズはATRに基づいて計算されます。

RangeMode 列挙体

ファイル `wijmo.chart.finance.js`
モジュール **wijmo.chart.finance**

カギ足および練行足チャートタイプの単位を指定します。

メンバー

名前	値	説明
Fixed	0	カギ足チャートの反転量または練行足チャートのボックスサイズに、正の固定値を使用します。
ATR	1	カギ足チャートの反転量または練行足チャートのボックスサイズに、現在のATR（Average True Range）値を使用します。 ATRを使用する場合は、これらのチャートの反転量またはボックスサイズオプションを整数にする必要があります。これらは、ATR計算の期間として使用されます。
Percentage	2	カギ足チャートの反転量に割合を使用します。 現在、練行足チャートタイプはRangeMode.Percentageをサポートしていません。

wijmo.chart.finance.analytics モジュール

ファイル `wijmo.chart.finance.analytics.js`
モジュール **wijmo.chart.finance.analytics**

FinancialChart の分析拡張機能です。

クラス

- ATR
- BollingerBands
- CCI
- Envelopes
- Fibonacci
- FibonacciArcs
- FibonacciFans
- FibonacciTimeZones
- Macd
- MacdBase
- MacdHistogram
- OverlayIndicatorBase
- RSI
- SingleOverlayIndicatorBase
- Stochastic
- WilliamsR

ATR クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**SingleOverlayIndicatorBase**
派生クラス	**WjFlexChartAtr**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用のアベレージトゥルーレンジインジケーター系列を表します。

アベレージトゥルーレンジは資産の変動率の測定に使用されます。アベレージトゥルーレンジは値動きの兆候を示すものではなく、価格がどの程度変動しているかを示します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): ATR
```

ATR クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **ATR**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

計算の期間を整数値として取得または設定します。

継承元 **SingleOverlayIndicatorBase**
型 **any**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

BollingerBands クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**OverlayIndicatorBase**
派生クラス	**WjFlexChartBollingerBands**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用のボリンジャーバンド（Bollinger Bands®）オーバーレイ系列を表します。

Bollinger BandsはJohn Bollinger氏の登録商標です。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- multiplier
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): BollingerBands
```

BollingerBands クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **BollingerBands**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

multiplier

標準偏差乗数を取得または設定します。

型 **number**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

period

計算の期間を整数値として取得または設定します。

型 **any**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

CCI クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**SingleOverlayIndicatorBase**
派生クラス	**WjFlexChartCci**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用の商品チャネル指数インジケーター系列を表します。

商品チャネル指数は、資産の現在の価格水準を指定期間の平均価格水準に対して測定するオシレーターです。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- constant
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): CCI
```

CCI クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **CCI**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- constant

CCI計算の定数値を取得または設定します。デフォルト値は0.015です。

型 **number**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

period

計算の期間を整数値として取得または設定します。

継承元 **SingleOverlayIndicatorBase**
型 **any**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

Envelopes クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**OverlayIndicatorBase**
派生クラス	**WjFlexChartEnvelopes**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用の移動平均エンベロープオーバーレイ系列を表します。

移動平均エンベロープは、標準移動平均の上下に設定された移動平均です。標準移動平均の上下の量はパーセントベースであり、 **size** プロパティによって指定されます。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- period
- size
- style
- symbolMarker
- symbolSize
- symbolStyle
- type
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Envelopes
```

Envelopes クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Envelopes**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 `SeriesBase`
型 **string**

period

計算の期間を整数値として取得または設定します。

型 **any**

size

移動平均エンベロープのサイズを取得または設定します。デフォルト値は2.5%（0.025）です。

型 **number**

style

系列のスタイルを取得または設定します。

継承元 `SeriesBase`
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 `SeriesBase`
型 `Marker`

symbolSize

この`Series`をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 `SeriesBase`
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 `SeriesBase`
型 **any**

type

エンベロープの移動平均の種類を取得または設定します。デフォルト値はSimpleです。

型 **MovingAverageType**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

Fibonacci クラス

ファイル	wijmo.chart.finance.analytics.js
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**SeriesBase**
派生クラス	**WjFlexChartFibonacci**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用のフィボナッチアークツールを表します。 このツールを使用すると、金融チャートに役立つさまざまなアラートレベルを計算してプロットできます。

フィボナッチツールを**FinancialChart** コントロールに追加するには、**Fibonacci** のインスタンスを作成し、それをチャートの**series**コレクションに追加します。 例:

```
// チャートを作成します。
var chart = new wijmo.chart.finance.FinancialChart('#chartElement');

// フィボナッチツールを作成します。
var ftool = new wijmo.chart.finance.analytics.Fibonacci();
chart.series.push(ftool);
```

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- high
- hostElement
- interpolateNulls
- itemsSource
- labelPosition
- legendElement
- levels
- low
- maxX
- minX
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- uptrend
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Fibonacci
```

Fibonacci クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Fibonacci**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

high

Fibonacci ツールの高値を取得または設定します。

これを指定しない場合、高値は**itemsSource** によって提供されたデータ値に基づいて計算されます。

型 **number**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- labelPosition

Fibonacci ツールのレベルのラベル位置を取得または設定します。

型 **LabelPosition**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- levels

プロットに使用する水準の配列を取得または設定します。

デフォルト値は[0, 23.6, 38.2, 50, 61.8, 100]です。

型 **number[]**

- low

Fibonacci ツールの安値を取得または設定します。

これを指定しない場合、安値は**itemsSource**によって提供されたデータ値に基づいて計算されます。

型 **number**

- maxX

Fibonacci ツールのxの最大値を取得または設定します。

これを指定しない場合、x軸の現在の最大値が使用されます。 値は数値またはDateオブジェクトで指定できます。

型 **any**

- minX

Fibonacci ツールのxの最小値を取得または設定します。

これを指定しない場合、x軸の現在の最小値が使用されます。 値は数値またはDateオブジェクトで指定できます。

型 **any**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

uptrend

上昇トレンドの**Fibonacci** ツールを作成するかどうかを示す値を取得または設定します。

デフォルト値はtrue（上昇トレンド）です。この値がfalseの場合は、下降トレンドのレベルがプロットされます。

型 **boolean**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

FibonacciArcs クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**SeriesBase**
派生クラス	**WjFlexChartFibonacciArcs**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用のフィボナッチアークツールを表します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- end
- hostElement
- interpolateNulls
- itemsSource
- labelPosition
- legendElement
- levels
- name
- start
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): FibonacciArcs
```

FibonacciArcs クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  初期化データを含むJavaScriptオブジェクト。

戻り値 **FibonacciArcs**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

end

ベースラインの終点の **DataPoint** を取得または設定します。

DataPoint のxの値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

他の一部のフィボナッチツールとは異なり、終点の**DataPoint** を定義しなかった場合、これは自動的に計算**not**。

型 **DataPoint**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- labelPosition

FibonacciArcs ツールの水準の**LabelPosition** を取得または設定します。

型 **LabelPosition**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- levels

プロットに使用する水準の配列を取得または設定します。

デフォルト値は[38.2, 50, 61.8]です。

型 **number**[]

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- start

ベースラインの始点の **DataPoint** を取得または設定します。

DataPoint のxの値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

他の一部のフィボナッチツールとは異なり、始点の**DataPointを定義しなかった場合**、これは自動的に計算**not**。

型 **DataPoint**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

このSeries をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 SeriesBase
型 number

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 SeriesBase
型 any

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 SeriesBase
型 SeriesVisibility

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標のPoint 。

継承元 SeriesBase
戻り値 Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 SeriesBase
戻り値 void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

FibonacciFans クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**SeriesBase**
派生クラス	**WjFlexChartFibonacciFans**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用のフィボナッチファンツールを表します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- end
- hostElement
- interpolateNulls
- itemsSource
- labelPosition
- legendElement
- levels
- name
- start
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): FibonacciFans
```

FibonacciFans クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  初期化データを含むJavaScriptオブジェクト。

戻り値 **FibonacciFans**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

end

ベースラインの終点の **DataPoint** を取得または設定します。

設定されていない場合、始点の**DataPoint** は自動的に計算されます。 **DataPoint** のxの値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

型 **DataPoint**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

labelPosition

FibonacciFans ツールの水準の**LabelPosition** を取得または設定します。

型 **LabelPosition**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

levels

プロットに使用する水準の配列を取得または設定します。

デフォルト値は[0, 23.6, 38.2, 50, 61.8, 100]です。

型 **number**[]

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

start

ベースラインの始点の **DataPoint** を取得または設定します。

設定されていない場合、始点の**DataPoint** は自動的に計算されます。 **DataPoint** のxの値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

型 **DataPoint**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

このSeries をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 SeriesBase
型 number

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 SeriesBase
型 any

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 SeriesBase
型 SeriesVisibility

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標のPoint 。

継承元 SeriesBase
戻り値 Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 SeriesBase
戻り値 void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

FibonacciTimeZones クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**SeriesBase**
派生クラス	**WjFlexChartFibonacciTimeZones**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用のフィボナッチタイムゾーンツールを表します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- endX
- hostElement
- interpolateNulls
- itemsSource
- labelPosition
- legendElement
- levels
- name
- startX
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): FibonacciTimeZones
```

FibonacciTimeZones クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  初期化データを含むJavaScriptオブジェクト。

戻り値 **FibonacciTimeZones**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

endX

タイムゾーンの終了位置のXデータポイントを取得または設定します。

設定されていない場合、終了位置のXデータポイントは自動的に計算されます。値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

型 **any**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- labelPosition

FibonacciTimeZones ツールの水準の**LabelPosition** を取得または設定します。

型 **LabelPosition**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- levels

プロットに使用する水準の配列を取得または設定します。

デフォルト値は[0, 1, 2, 3, 5, 8, 13, 21, 34]です。

型 **number**[]

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- startX

タイムゾーンの開始位置のXデータポイントを取得または設定します。

設定されていない場合、開始位置のXデータポイントは自動的に計算されます。値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

型 **any**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

このSeries をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 SeriesBase
型 number

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 SeriesBase
型 any

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 SeriesBase
型 SeriesVisibility

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標のPoint 。

継承元 SeriesBase
戻り値 Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 SeriesBase
戻り値 void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

Macd クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**MacdBase**
派生クラス	**WjFlexChartMacd**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用の移動平均収束拡散（MACD）インジケーター系列を表します。

MACDインジケーターは、資産の値動きの強さ、方向、モメンタム、および持続期間の変化を表します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- fastPeriod
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- slowPeriod
- smoothingPeriod
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Macd
```

Macd クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Macd**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

fastPeriod

MACDラインの高速の指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

slowPeriod

MACDラインの低速の指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

smoothingPeriod

シグナルラインの指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

styles

MACDラインとシグナルラインのスタイルを取得または設定します。

以下のオプションがサポートされています。

```
series.styles = {
  macdLine: {
     stroke: 'red',
     strokeWidth: 1
  },
  signalLine: {
     stroke: 'green',
     strokeWidth: 1
  },
}
```

型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

MacdBase クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**OverlayIndicatorBase**
派生クラス	**Macd**, **MacdHistogram**
表示	☐継承されたメンバー ☐イベント発生元

Macd および**MacdHistogram** 系列の基本クラス（抽象）。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- fastPeriod
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- slowPeriod
- smoothingPeriod
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): MacdBase
```

MacdBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **MacdBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- fastPeriod

MACDラインの高速の指数移動平均の期間を取得または設定します。

型 **number**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

slowPeriod

MACDラインの低速の指数移動平均の期間を取得または設定します。

型 **number**

smoothingPeriod

シグナルラインの指数移動平均の期間を取得または設定します。

型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

MacdHistogram クラス

ファイル	wijmo.chart.finance.analytics.js
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**MacdBase**
派生クラス	**WjFlexChartMacdHistogram**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用の移動平均収束拡散（MACD）ヒストグラムインジケーター系列を表します。

MACDインジケーターは、資産の値動きの強さ、方向、モメンタム、および持続期間の変化を表します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- fastPeriod
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- slowPeriod
- smoothingPeriod
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): MacdHistogram
```

MacdHistogram クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **MacdHistogram**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

fastPeriod

MACDラインの高速の指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

slowPeriod

MACDラインの低速の指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

smoothingPeriod

シグナルラインの指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

OverlayIndicatorBase クラス

ファイル wijmo.chart.finance.analytics.js
モジュール **wijmo.chart.finance.analytics**
基本クラス **SeriesBase**
派生クラス **BollingerBands**, **Envelopes**, **MacdBase**, **SingleOverlayIndicatorBase**, **Stochastic**
表示 ☐継承されたメンバー ☐イベント発生元

オーバーレイおよびインジケーター系列の基本クラス（抽象）。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): OverlayIndicatorBase
```

OverlayIndicatorBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **OverlayIndicatorBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

RSI クラス

ファイル wijmo.chart.finance.analytics.js
モジュール **wijmo.chart.finance.analytics**
基本クラス **SingleOverlayIndicatorBase**
派生クラス **WjFlexChartRsi**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart 用の相対力指数インジケーター系列を表します。

相対力指数は、資産の現在および過去の強さまたは弱さを最近の取引期間の終値に基づいて測定するために設計されたモメンタムオシレーターです。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): RSI
```

RSI クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **RSI**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

計算の期間を整数値として取得または設定します。

継承元 **SingleOverlayIndicatorBase**
型 **any**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

SingleOverlayIndicatorBase クラス

ファイル	wijmo.chart.finance.analytics.js
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**OverlayIndicatorBase**
派生クラス	**ATR**, **CCI**, **RSI**, **WilliamsR**
表示	☐継承されたメンバー ☐イベント発生元

単一の系列をレンダリングするオーバーレイおよびインジケーター系列の基本クラス（抽象）。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): SingleOverlayIndicatorBase
```

SingleOverlayIndicatorBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **SingleOverlayIndicatorBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

period

計算の期間を整数値として取得または設定します。

型 **any**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

Stochastic クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**OverlayIndicatorBase**
派生クラス	**WjFlexChartStochastic**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用のストキャスティクスオシレーターインジケーター系列を表します。

ストキャスティクスは、資産の終値を高値安値レンジと比較することによって価格転換点を予測するために設計されたモメンタムインジケーターです。

Stochastic 系列は、ファストストキャスティクス（デフォルト）、スローストキャスティクス、およびフルストキャスティクスに使用できます。スローストキャスティクスまたはフルストキャスティクスを作成するには、 **smoothingPeriod** を1より大きい整数値に設定します。スローストキャスティクスには通常、決められた**smoothingPeriod** （3）を使用します。ファストストキャスティクスを作成するか、ファストストキャスティクスに戻すには、**smoothingPeriod** を1に設定します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- dPeriod
- hostElement
- interpolateNulls
- itemsSource
- kPeriod
- legendElement
- name
- smoothingPeriod
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Stochastic
```

Stochastic クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **Stochastic**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

dPeriod

%D単純移動平均の期間を取得または設定します。

型 **number**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

kPeriod

%K計算の期間を取得または設定します。

型 **number**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

smoothingPeriod

フル%Kの平滑化期間を取得または設定します。

型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

styles

%Kラインと%Dラインのスタイルを取得または設定します。

以下のオプションがサポートされています。

```
series.styles = {
  kLine: {
     stroke: 'red',
     strokeWidth: 1
  },
  dLine: {
     stroke: 'green',
     strokeWidth: 1
  },
}
```

型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

WilliamsR クラス

ファイル	`wijmo.chart.finance.analytics.js`
モジュール	**wijmo.chart.finance.analytics**
基本クラス	**SingleOverlayIndicatorBase**
派生クラス	**WjFlexChartWilliamsR**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart 用のウィリアムズ%Rインジケーター系列を表します。

ウィリアムズ%Rは、ファストストキャスティクス（ **Stochastic**）を逆転させたモメンタム指標です。ウィリアムズ%Rインジケーターは、資産の取引が取引レンジの高値に近い方でなされているか、安値に近い方でなされているかを示します。

コンストラクタ

- constructor

プロパティ

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- interpolateNulls
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

メソッド

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): WilliamsR
```

WilliamsR クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

戻り値 **WilliamsR**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

計算の期間を整数値として取得または設定します。

継承元 **SingleOverlayIndicatorBase**
型 **any**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

メソッド

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

wijmo.olap モジュール

ファイル `wijmo.olap.js`
モジュール **wijmo.olap**

ピボットテーブル、チャートなどのOLAP機能を提供するコンポーネントが含まれます。

PivotEngine クラスは、生データを集計してピボットビューに 表示します。

PivotPanel コントロールは、フィールドをビューリストに ドラッグし、そのプロパティを編集することで、ピボットビューを 編集するためのUIを提供します。

PivotGrid コントロールは、**FlexGrid** を拡張して、 折りたたみ可能な行および列のグループを含む ピボットテーブルを表示します。

PivotChart は、階層的な軸を含むピボットテーブルの ビジュアル表現を提供します。

クラス

- CubePivotField
- DetailDialog
- PivotChart
- PivotCollectionView
- PivotEngine
- PivotField
- PivotFieldCollection
- PivotFieldEditor
- PivotFilter
- PivotFilterEditor
- PivotGrid
- PivotPanel
- ProgressEventArgs
- Slicer

列挙体

- DimensionType
- LegendVisibility
- PivotChartType
- ShowAs
- ShowTotals

CubePivotField クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**
基本クラス **PivotField**
表示 ☐継承されたメンバー ☐イベント発生元

PivotField クラスを拡張して、サーバーベースのキューブデータソース内の フィールドを表します。

コンストラクタ

- constructor

プロパティ

- aggregate
- binding
- collectionView
- dataType
- descending
- dimensionType
- engine
- filter
- format
- getValue
- header
- isActive
- isContentHtml
- isMeasure
- key
- parentField
- showAs
- sortComparer
- subFields
- visible
- weightField
- width
- wordWrap

メソッド

- onPropertyChanged

イベント

- propertyChanged

コンストラクタ

constructor

```
constructor(engine: PivotEngine, binding: string, header?: string, options?: any): CubePivotField
```

CubePivotField クラスの新しいインスタンスを初期化します。

パラメーター

- **engine**: **PivotEngine**
  このフィールドを所有する**PivotEngine** 。
- **binding**: **string**
  このフィールドの連結先のプロパティ。
- **header**: **string** OPTIONAL
  このフィールドを識別するために表示されるヘッダー（デフォルトは連結先と同じ）。
- **options**: **any** OPTIONAL
  フィールドの初期化データを含むJavaScriptオブジェクト。

戻り値 **CubePivotField**

プロパティ

aggregate

Gets or sets how the field should be summarized.

The default value for this property is **Aggregate.Sum** for numeric fields, and **Aggregate.Count** for other field types.

継承元 **PivotField**
型 **Aggregate**

binding

フィールドの連結先のプロパティの名前を取得または設定します。

継承元 **PivotField**
型 **string**

collectionView

このフィールドに連結された**ICollectionView** を取得します。

継承元 **PivotField**
型 **ICollectionView**

dataType

フィールドのデータ型を取得または設定します。

継承元 **PivotField**
型 **DataType**

- descending

このフィールドのキーを降順でソートするかどうかを決定する値を取得または設定します。

The default value for this property is **false**.

継承元 **PivotField**
型 **boolean**

- dimensionType

フィールドのディメンションタイプを取得または設定します。

型 **DimensionType**

- engine

この**PivotField** を所有する**PivotEngine** への参照を取得します。

継承元 **PivotField**
型 **PivotEngine**

- filter

このフィールドの値をフィルタ処理するために使用される**PivotFilter** への参照を取得します。

キューブデータソースでのメジャーフィールドの場合、フィルターは集計セル値に適用されます。ノンキューブデータソースでのメジャーフィールドの場合、フィルターは生データに適用されます。

継承元 **PivotField**
型 **PivotFilter**

- format

フィールド値の表示に使用する書式を取得または設定します。

The default value for this property is **'d'** for date fields, **'n0'** for numeric fields, and the empty string for other field types.

継承元 **PivotField**
型 **string**

- getValue

Gets or sets a function to be used for retrieving the field values.

This property is set to null by default, causing the engine to use the field's **binding** property to retrieve the value.

If specified, the function should take a single parameter that represents the data item being evaluated and should return the calculated value for the item.

継承元 **PivotField**
型 **Function**

- header

ユーザーインタフェースでこのフィールドを表すために使用される文字列を取得または設定します。

型 **string**

- isActive

このフィールドが現在ビューで使用されているかどうかを判定する値を 取得または設定します。

このプロパティをtrueに設定すると、フィールドのデータ型に応じて ビューの**rowFields** または**valueFields** に フィールドが追加されます。

継承元 **PivotField**
型 **boolean**

- isContentHtml

このフィールドの項目がプレーンテキストではなく、HTMLコンテンツを 含むかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **PivotField**
型 **boolean**

- isMeasure

ディメンションの種類を考慮してオーバーライドされます。

型 **boolean**

- key

この**CubePivotField** のキーを取得します。

このタイプのフィールドの場合、キーはフィールドの**binding** です。

型 **string**

- parentField

フィールドの親フィールドを取得します。

同じフィールドを［値］リストに複数回ドラッグすると、 フィールドのコピーが作成され、同じ連結を異なるパラメータで 使用できます。コピーは、その親フィールドへの参照を保持します。

継承元 **PivotField**
型 **PivotField**

- showAs

Gets or sets a value that specifies how to display the aggregate value.

Options for this property are defined by the **ShowAs** enumeration and include differences between the value and the one in the previous row or column, percentages over the row, column, or grand total, and running totals.

This property is similar to the **Show Values As** feature in Excel.

The default value for this property is **ShowAs.NoCalculation**.

継承元 **PivotField**
型 **ShowAs**

sortComparer

ソート時に値の比較に使用する関数を取得または設定します。

指定された場合、ソート比較関数は、パラメータとして任意の型の値を 2つ取り、最初の値が2番目の値と比べて小さい、等しい、または大きい のいずれであるかを示す値-1、0、または+1を返します。ソート比較関数がnullを返す場合は、標準の組み込み 比較子が使用されます。

この**sortComparer** プロパティを使用すると、カスタム比較アルゴリズムを 提供でき、単純な文字列比較より、ユーザーが期待する結果によく 一致するソートシーケンスが得られる場合があります。

次の例は、**sortComparer** プロパティの一般的な使用方法を示します。

```
/// 製品のリストを定義します
app.products = 'Wijmo,Aoba,Olap,Xuni'.split(',');

// 'app.products'配列の位置で商品をソートします
ng.viewDefinitionChanged.addHandler(function () {
  var fld = ng.fields.getField('Product');
  if (fld) {
    fld.sortComparer = function (val1, val2) {
      return app.products.indexOf(val1) - app.products.indexOf(val2);
    }
  }
});
```

継承元 **PivotField**
型 **Function**

subFields

このフィールドの子フィールドを取得します。

継承元 **PivotField**
型 **PivotField**[]

visible

Gets or sets a value indicating whether this field should be displayed in instances of the **PivotPanel** control.

The default value for this property is **true**.

Setting this property to false hides the field any **PivotPanel** controls, preventing users from adding, removing, or changing the the field position in the engine's view definition.

継承元 **PivotField**
型 **boolean**

weightField

このフィールドの集計の計算時に重みとして使用する**PivotField** を 取得または設定します。

このプロパティがnullに設定されている場合は、すべての値が重み1と見なされます。

このプロパティを使用すると、加重平均や加重合計を計算できます。 たとえば、データに'Quantity'フィールドと'Price'フィールドがある場合に、'Price' フィールドを値フィールドとして使用し、'Quantity'フィールドを 重みとして使用できます。出力には、データの加重平均が含まれます。

継承元 **PivotField**
型 **PivotField**

width

このフィールドを**PivotGrid** のようなユーザーインタフェースコントロールで表示するために使用する適切な幅を取得または設定します。

継承元 **PivotField**
型 **number**

wordWrap

このフィールドのコンテンツをセル内で折り返すことができるかどうかを示す値を 取得または設定します。

The default value for this property is **false**.

継承元 **PivotField**
型 **boolean**

メソッド

onPropertyChanged

```
onPropertyChanged(e: PropertyChangedEventArgs): void
```

propertyChanged イベントを発生させます。

パラメーター

- **e**: **PropertyChangedEventArgs**
  プロパティ名および新旧の値を含む**PropertyChangedEventArgs** 。

継承元 **PivotField**
戻り値 **void**

イベント

propertyChanged

この**Range** のプロパティ値が変更されると発生します。

継承元 **PivotField**
引数 **PropertyChangedEventArgs**

DetailDialog クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**
基本クラス **Popup**
表示 ☐継承されたメンバー ☐イベント発生元

グリッドセルの詳細の表示に使用されるダイアログを表します。

コンストラクタ

- constructor

プロパティ

- cellHeader
- cellValue
- columnHeader
- content
- controlTemplate
- detailCount
- dialogResult
- dialogResultEnter
- fadeIn
- fadeOut
- hideTrigger
- hostElement
- isDisabled
- isDraggable
- isTouching
- isUpdating
- isVisible
- modal
- owner
- removeOnHide
- rightToLeft
- rowHeader
- showTrigger

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hide

- initialize
- invalidate
- invalidateAll
- onGotFocus
- onHidden
- onHiding
- onLostFocus
- onRefreshed
- onRefreshing
- onShowing
- onShown
- refresh
- refreshAll
- removeEventListener
- show

イベント

- gotFocus
- hidden
- hiding
- lostFocus
- refreshed
- refreshing
- showing
- shown

コンストラクタ

constructor

```
constructor(element: any, options?): DetailDialog
```

DetailDialog クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **DetailDialog**

プロパティ

- cellHeader

Gets the cell header for the value being shown.

This information is updated before the dialog is shown and is displayed above the detail grid.

型 **string**

cellValue

Gets the formatted cell value for the value being shown.

This information is updated before the dialog is shown and is displayed above the detail grid.

型 **string**

columnHeader

Gets the column header for the value being shown.

This information is updated before the dialog is shown and is displayed above the detail grid.

型 **string**

content

この**Popup** に含まれるHTML要素を取得または設定します。

継承元 **Popup**
型 **HTMLElement**

STATIC controlTemplate

PivotFieldEditor コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

detailCount

Gets the number of items shown in the detail dialog.

This information is updated before the dialog is shown and is in the dialog header.

型 **number**

dialogResult

Popup を非表示にした後に、そのコンテンツを処理するために使用できる値を取得または設定します。

このプロパティは、**Popup** が表示されたときにnullに設定され、ボタンクリックイベントに 応答して、または**hide** メソッドの呼び出しの中で設定できます。

継承元 **Popup**
型 **any**

dialogResultEnter

Popup 表示中にユーザーが［Enter］キーを押したときに**dialogResult** として 使用される値を取得または設定します。

ユーザーが［Enter］を押し、**dialogResultEnter** プロパティがnullでない場合、 ポップアップは、すべての子要素が有効な状態かどうかをチェックします。 有効な状態の場合は、ポップアップが閉じ、**dialogResult** プロパティが **dialogResultEnter** プロパティの値に設定されます。

継承元 **Popup**
型 **any**

fadeIn

Popup を表示するときにフェードアウトアニメーションを使用する かどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **Popup**
型 **boolean**

fadeOut

Popup を非表示にするときにフェードアウトアニメーションを使用する かどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **Popup**
型 **boolean**

hideTrigger

ポップアップを非表示にするアクションを取得または設定します。

デフォルトでは、**showTrigger** プロパティは**Blur** に設定されており、 フォーカスを失った場合、またはEscキーを押した時にポップアップが非表示になります。

hideTrigger プロパティを**Click** に設定すると、ポップアップはオーナ要素を クリックする場合、またはEscキーを押した時非表示になります。

hideTrigger プロパティを**None** に設定すると、Popupは**hide** メソッドを 呼び出します。

継承元 **Popup**
型 **PopupTrigger**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isDraggable

ポップアップをそのヘッダーでマウスによってドラッグできるかどうかを示す値を取得または設定します。

ヘッダーは '.wj-dialog-header' のCSSセレクターによって識別されます。 ダイアログに 'wj-dialog-header'クラスの要素が含まれていない場合、ユーザーはポップアップをドラッグできません。

ポップアップをドラッグ可能にする場合は、 '.wj-dialog-header' CSSセレクタのcursorプロパティを設定することができます。次に例を示しています。

```
<style>
  .wj-popup {
    width: 30%;
  }
  .wj-dialog-header {
    cursor: move;
  }
</style>
```

The default value for this property is **false**.

継承元 **Popup**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isVisible

Popupが現在表示されているかどうかを決定する値を取得します。

継承元 **Popup**
型 **boolean**

modal

Popup をモーダルダイアログとして表示するかどうかを決定する値を 取得または設定します。

モーダルダイアログは、 **Popup** がページ内の他のコンテンツより目立つように、 暗い背景が付けられます。

ダイアログを完全にモーダルにするには、 **hideTrigger** プロパティを **None** に設定して、ユーザーが背景をクリックしても ダイアログを閉じることができないようにします。この場合は、**hide** メソッドが呼び出されるか、 ユーザーが［Esc］キーを押した場合にのみ、ダイアログが閉じられます。

The default value for this property is **false**.

継承元 **Popup**
型 **boolean**

- owner

Popup を所有する要素を取得または設定します。オーナーがnullの場合、ポップアップはダイアログとして動作します。ダイアログは、画面の中央に配置され、**show** メソッドを使用して表示する必要があります。

継承元 **Popup**
型 **HTMLElement**

- removeOnHide

Popup が非表示になっているときに**Popup** 要素をDOMから削除するか非表示にするかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **Popup**
型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- rowHeader

Gets the row header for the value being shown.

This information is updated before the dialog is shown and is displayed above the detail grid.

型 **string**

- showTrigger

Popup を表示するアクションを取得または設定します。デフォルトでは、 **showTrigger** プロパティは**Click** に設定されており、オーナー要素をクリックすると、ポップアップが表示されます。 **showTrigger** プロパティを**None** に設定すると、ポップアップは **show** メソッドを呼び出すのみで表示されます。

継承元 **Popup**
型 **PopupTrigger**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hide

```
hide(dialogResult?: any): void
```

Popup を非表示にします。

パラメーター

- **dialogResult**: **any** OPTIONAL
  Popup を閉じる前に**dialogResult** プロパティに割り当てられる オプションの値。

継承元 **Popup**
戻り値 **void**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onHidden

```
onHidden(e?: EventArgs): void
```

hidden イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Popup**
戻り値 **void**

onHiding

```
onHiding(e: CancelEventArgs): boolean
```

hiding イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **Popup**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onShowing

```
onShowing(e: CancelEventArgs): boolean
```

showing イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **Popup**
戻り値 **boolean**

onShown

```
onShown(e?: EventArgs): void
```

shown イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Popup**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

show

```
show(modal?: boolean, handleResult?: Function): void
```

Popup を表示します。

パラメーター

- **modal**: **boolean** OPTIONAL
  ポップアップをモーダルダイアログとして表示するかどうか。 指定した場合、その値が **modalプロパティに設定されます**。

- **handleResult**: **Function** OPTIONAL
  ポップアップが非表示になると、コールバックが呼び出されます。 コールバックが指定されている場合は、パラメータとしてポップアップを受け取ります。

  handleResultコールバックを使用すると、**hidden** イベントにハンドラを アタッチしなくても、呼び出し元がモーダルダイアログの結果を処理できます。 たとえば、 以下のコードは、**CollectionView** 内の現在の項目を編集するための ダイアログを表示します。 編集結果は、**dialogResult** 値に応じてコミット またはキャンセルされます。 次に例を示します。

```
$scope.editCurrentItem = function () {
  $scope.data.editItem($scope.data.currentItem);
  $scope.itemEditor.show(true, function (e) {
    if (e.dialogResult == 'wj-hide-ok') {
      $scope.data.commitEdit();
    } else {
      $scope.data.cancelEdit();
    }
  });
}
```

継承元 **Popup**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

hidden

Popup が非表示になる後に発生します。

継承元 **Popup**
引数 **EventArgs**

hiding

Popup が非表示になる前に発生します。

継承元 **Popup**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

showing

Popup が表示される前に発生させます。

継承元 **Popup**
引数 **CancelEventArgs**

shown

ポップアップが表示された後に発生させます。

継承元 **Popup**
引数 **EventArgs**

PivotChart クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**
基本クラス **Control**
派生クラス **WjPivotChart**
表示 ☐継承されたメンバー ☐イベント発生元

wijmo.olap ピボットテーブルのビジュアル表現を提供します。

このコントロールを使用するには、その**itemsSource** プロパティを **PivotPanel** コントロールのインスタンスまたは**PivotEngine** に設定します。

コンストラクタ

- constructor

プロパティ

- chartType
- engine
- flexChart
- flexPie
- footer
- footerStyle
- header
- headerStyle
- hostElement
- isDisabled
- isTouching
- isUpdating
- itemsSource
- legendPosition
- maxPoints
- maxSeries
- rightToLeft
- showHierarchicalAxes
- showLegend
- showTitle
- showTotals
- stacking

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate

- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, options?): PivotChart
```

PivotChart クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **PivotChart**

プロパティ

- chartType

作成するチャートのタイプを取得または設定します。

The default value for this property is **PivotChartType.Column**.

型 **PivotChartType**

- engine

この**PivotChart** を所有する**PivotEngine** への参照を取得します。

型 **PivotEngine**

- flexChart

内部の**FlexChart**コントロールへの参照を取得します。

型 `FlexChart`

- flexPie

内部の**FlexPie**コントロールへの参照を取得します。

型 `FlexPie`

- footer

チャートのフッタに表示されるテキストを取得または設定します。

型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

型 **any**

- header

チャートのヘッダに表示されるテキストを取得または設定します。

型 **string**

- headerStyle

チャートのヘッダスタイルを取得または設定します。

型 **any**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

itemsSource

この`PivotChart` のデータを提供する`PivotEngine` または`PivotPanel` を 取得または設定します。

型 **any**

legendPosition

凡例を表示するかどうか、表示する場合はプロットエリアに対してどの位置に表示するかを決定する値を取得または設定します。

The default value for this property is **Position.Right**.

型 `Position`

maxPoints

各系列に表示するポイントの最大数を取得または設定します。

The default value for this property is **100** points.

型 **number**

maxSeries

チャートに表示するデータ系列の最大数を取得または設定します。

The default value for this property is **100** series.

型 **number**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- showHierarchicalAxes

グループ化されたデータに対してチャートの軸注釈をグループ化するかどうかを決定する値を　取得または設定します。

The default value for this property is **true**.

型　　**boolean**

- showLegend

チャートに凡例を含めるかどうかを決定する値を取得または設定します。

The default value for this property is **LegendVisibility.Always**.

型　　**`LegendVisibility`**

- showTitle

チャートにタイトルを含めるかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型　　**boolean**

- showTotals

チャートに合計だけを含めるかどうかを決定する値を取得または設定します。

If showTotals is true and the view has Column Fields, then the chart will show column totals instead of individual values.

The default value for this property is **false**.

型　　**boolean**

- stacking

系列オブジェクトを積み重ねるかどうかを決定する値と、積み重ねる場合はその方法を取得または設定します。

The default value for this property is **Stacking.None**.

型　　**`Stacking`**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**ImageFormat** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

PivotCollectionView クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**
基本クラス **CollectionView**
表示 ☐継承されたメンバー ☐イベント発生元

CollectionView クラスを拡張して、ソート時に小計行の位置を維持します。

コンストラクタ

- constructor

プロパティ

- canAddNew
- canCancelEdit
- canChangePage
- canFilter
- canGroup
- canRemove
- canSort
- currentAddItem
- currentEditItem
- currentItem
- currentPosition
- engine
- filter
- getError
- groupDescriptions
- groups
- isAddingNew
- isEditingItem
- isEmpty
- isPageChanging
- isUpdating
- itemCount
- items
- itemsAdded
- itemsEdited
- itemsRemoved
- newItemCreator
- pageCount
- pageIndex
- pageSize
- sortComparer
- sortConverter
- sortDescriptions
- sortNullsFirst
- sourceCollection
- totalItemCount
- trackChanges

- useStableSort

メソッド

- addNew
- beginUpdate
- cancelEdit
- cancelNew
- clearChanges
- commitEdit
- commitNew
- contains
- deferUpdate
- editItem
- endUpdate
- getAggregate
- implementsInterface
- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage
- onCollectionChanged
- onCurrentChanged
- onCurrentChanging
- onPageChanged
- onPageChanging
- onSourceCollectionChanged
- onSourceCollectionChanging
- refresh
- remove
- removeAt

イベント

- collectionChanged
- currentChanged
- currentChanging
- pageChanged
- pageChanging
- sourceCollectionChanged
- sourceCollectionChanging

コンストラクタ

constructor

```
constructor(engine: PivotEngine): PivotCollectionView
```

PivotCollectionView クラスの新しいインスタンスを初期化します。

パラメーター

- **engine**: **PivotEngine**
  このコレクションを所有する**PivotEngine** 。

戻り値 **PivotCollectionView**

プロパティ

canAddNew

コレクションに新しい項目を追加できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canCancelEdit

適用前の変更を破棄して編集されたオブジェクトの元の値を復元できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canChangePage

pageIndex 値を変更できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canFilter

このビューが**filter** プロパティによってフィルタリングをサポートしているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canGroup

このビューが**groupDescriptions** プロパティによってグループ化をサポートしているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canRemove

コレクションから項目を削除できるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

canSort

このビューが**sortDescriptions** プロパティによってソートをサポートしているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

currentAddItem

現在の追加トランザクションの間に追加される項目を取得します。

継承元 **CollectionView**
型 **any**

currentEditItem

現在の編集トランザクションの間に編集される項目を取得します。

継承元 **CollectionView**
型 **any**

currentItem

ビューの現在の項目を取得します。

継承元 **CollectionView**
型 **any**

currentPosition

ビューの現在の項目の順序位置を取得します。

継承元 **CollectionView**
型 **number**

engine

このビューを所有する**PivotEngine** への参照を取得します。

型 **PivotEngine**

filter

項目がビューに含める対象として適しているかどうかを判断するために使用されるコールバックを取得または設定します。

このコールバック関数がtrueを返した場合、パラメーターとして渡された項目はビューに含まれます。

メモ: フィルタ関数でスコープ（すなわち、有効な'this'値）が必要な場合は、'this'オブジェクトを指定した'bind'関数を使用してフィルタを設定します。例:

```
collectionView.filter = this._filter.bind(this);
```

継承元 **CollectionView**
型 **IPredicate**

getError

項目の特定のプロパティに検証エラーが含まれているかどうかを判定するコールバックを取得または設定します。

指定すると、コールバックは、検証する項目とプロパティを含む2つの パラメータを受け取り、エラーを説明する文字列を返します （エラーがない場合はnull）。

次に例を示します。

```
var view = new wijmo.collections.CollectionView(data, {
    getError: function (item, property) {
        switch (property) {
            case 'country':
                return countries.indexOf(item.country) < 0
                    ? 'Invalid Country'
                    : null;
            case 'downloads':
            case 'sales':
            case 'expenses':
                return item[property] < 0
                    ? '負にはできません。'
                    : null;
            case 'active':
                return item.active && item.country.match(/US|UK/)
                    ? 'USまたはUKではアクティブな項目が許可されません。'
                    : null;
        }
        return null;
    }
});
```

継承元 **CollectionView**
型 **Function**

groupDescriptions

コレクションの項目をビューでどのようにグループ化するかを記述する **GroupDescription** オブジェクトのコレクションを取得します。

継承元 **CollectionView**
型 **ObservableArray**

groups

最上位レベルのグループを表す **CollectionViewGroup** オブジェクトの配列を取得します。

継承元 **CollectionView**
型 **CollectionViewGroup**[]

isAddingNew

追加トランザクションが進行中であるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

isEditingItem

編集トランザクションが進行中であるかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isEmpty

このビューに項目が1つも含まれていないかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isPageChanging

ページインデックスが変更されているかどうかを示す値を取得します。

継承元 **CollectionView**
型 **boolean**

- isUpdating

通知が現在中断されているかどうかを示す値を取得します（**beginUpdate** および**endUpdate** を参照）。

継承元 **CollectionView**
型

- itemCount

ページングを適用する前のビューの既知の項目の数を取得します。

継承元 **CollectionView**
型 **number**

- items

ビューの項目を取得します。

継承元 **CollectionView**
型 **any**[]

- itemsAdded

trackChanges が有効化されてから、コレクションに追加されたレコードを含む**ObservableArray** を取得します。

継承元 **CollectionView**
型 **ObservableArray**

- itemsEdited

trackChanges が有効化されてから、コレクションで編集されたレコードを含む**ObservableArray** を取得します。

継承元 **CollectionView**
型 **ObservableArray**

- itemsRemoved

trackChanges が有効化されてから、コレクションから削除されたレコードを含む**ObservableArray** を取得します。

継承元 **CollectionView**
型 **ObservableArray**

newItemCreator

コレクションの新しい項目を作成する関数を取得または設定します。

作成関数が提供されない場合、**CollectionView** は、 適切な型の項目を初期化せずに作成しようとします。

作成関数が提供される場合、その関数は、パラメータを受け取らず、 コレクションに対して適切な型のオブジェクトを初期化して返す 必要があります。

継承元 **CollectionView**
型 **Function**

pageCount

総ページ数を取得します。

継承元 **CollectionView**
型 **number**

pageIndex

現在のページの0から始まるインデックスを取得します。

継承元 **CollectionView**
型 **number**

pageSize

1ページに表示する項目数を取得または設定します。

継承元 **CollectionView**
型 **number**

sortComparer

ソート時に値の比較に使用する関数を取得または設定します。

指定された場合、ソート比較関数は、パラメータとして任意の型の値を 2つ取り、最初の値が2番目の値と比べて小さい、等しい、または大きい のいずれであるかを示す値-1、0、または+1を返します。ソート比較関数がnullを返す場合は、標準の組み込み 比較子が使用されます。

この**sortComparer** プロパティを使用すると、カスタム比較アルゴリズムを 提供でき、単純な文字列比較より、ユーザーが期待する結果によく 一致するソートシーケンスが得られる場合があります。

たとえば、 **Dave Koeleの英数字アルゴリズム**があります。 このアルゴリズムは、文字列を文字列や数値から成る部分に分割した後、 数値部分は値順に、文字列部分はASCII順にソートします。 Daveは、この結果を「自然なソート順」と呼んでいます。

次の例は、**sortComparer** プロパティの一般的な使用方法を示します。

```
// カスタムソート比較子を使用してCollectionViewを作成します
var dataCustomSort = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b)
      ? alphanum(a, b) // 文字列に使用するカスタム比較子
      : null; // 文字列以外にはデフォルトの比較子を使用します
  }
});
```

次の例は、 **Intl.Collator** を使用してソート順を制御する方法を示しています。

```
// Intl.Collatorを使用してソートするCollectionViewを作成します
var collator = window.Intl ? new Intl.Collator() : null;
var dataCollator = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b) && collator
      ? collator.compare(a, b) // 文字列に使用するカスタム比較子
      : null; // 文字列以外にはデフォルトの比較子を使用します
  }
});
```

継承元 **CollectionView**
型 **Function**

sortConverter

ソート時の値の変換に使用される関数を取得または設定します。

この関数は、**SortDescription**、データ項目、および変換する値をパラメーターとして受け取り、変換後の値を返す必要があります。

このプロパティはソートの動作をカスタマイズする手段を提供します。たとえば、**FlexGrid** コントロールはこのプロパティを使用して、マップされた列を生の値ではなく表示値を基準にソートします。

以下のサンプルコードは、国コードの整数を含む'country'プロパティをソートするときに、 対応する国名を使用してソートされるようにします。

```
var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(',');
collectionView.sortConverter = function (sd, item, value) {
  if (sd.property == 'countryMapped') {
    value = countries[value]; // 国IDを国名に変換します。
  }
  return value;
}
```

継承元 **CollectionView**
型 **Function**

sortDescriptions

コレクションの項目をビューでどのようにソートするかを記述する**SortDescription** オブジェクトの配列を取得します。

継承元 **CollectionView**
型 **ObservableArray**

sortNullsFirst

Gets or sets a value that determines whether null values should appear first or last when the collection is sorted (regardless of sort direction).

This property is false by default, which causes null values to appear last on the sorted collection. This is also the default behavior in Excel.

継承元 **CollectionView**
型 **boolean**

sourceCollection

基になる（フィルタリングもソートもされていない）コレクションを取得または設定します。

継承元 **CollectionView**
型 **any**

totalItemCount

ページングを適用する前のビュー内の項目の合計数を取得します。

継承元 **CollectionView**
型 **number**

trackChanges

コントロールがデータの変更を追跡するかどうかを決定する値を取得または設定します。

trackChanges がtrueに設定されている場合、**CollectionView** は、データの変更を追跡し、 **itemsAdded**、**itemsRemoved**、**itemsEdited** の各コレクションを介して変更を公開します。

変更の追跡は、変更が有効であることをユーザーが確認した後にサーバーを更新する必要がある場合に役立ちます。

変更をコミットまたはキャンセルしたら、**clearChanges** メソッドを使用して、 **itemsAdded**、**itemsRemoved**、**itemsEdited** の各コレクションをクリアします。

CollectionView は、適切な**CollectionView** メソッド（**editItem** /**commitEdit**、 **addNew** /**commitNew**、**remove**）を使用して行われた変更だけを追跡します。 データに直接加えた変更は追跡されません。

継承元 **CollectionView**
型 **boolean**

useStableSort

安定したソートアルゴリズムを使用するかどうかを取得または設定します。

安定したソートアルゴリズムは、同じキーを持つレコード間の相対的な順序を維持します。 たとえば、"Amount"フィールドを持つオブジェクトのコレクションを考えてみます。 このコレクションを"Amount"でソートする場合、安定したソートでは、Amount値が同じレコード間で元の順序が保たれます。

このプロパティのデフォルトはfalseです。この場合は、高速だが安定ではない JavaScriptの組み込みソートメソッドが**CollectionView** で使用されます。**useStableSort** をtrueに設定すると、ソート時間が30%～50%も長くなります。 コレクションが大きいと、かなりの時間の増加になります。

継承元 **CollectionView**
型 **boolean**

メソッド

addNew

```
addNew(): any
```

新しい項目を作成し、コレクションに追加します。

このメソッドは、パラメータを受け取りません。新しい項目が**commitNew** メソッドでコミットされるか、 **cancelNew** メソッドでキャンセルされるまで、 リフレッシュ操作を保留します。

次のコードは、**addNew** メソッドの典型的な使用方法を示します。

```
// 新しい項目を作成し、それをコレクションに追加します
var newItem = view.addNew();

// 新しい項目を初期化します
newItem.id = getFreshId();
newItem.name = '新しい顧客';

// ビューをリフレッシュできるように新しい項目をコミットします
view.commitNew();
```

新しい項目を**sourceCollection** にプッシュしてから、**refresh** メソッドを 呼び出すことで、新しい項目を追加することもできます。**addNew** では、 ユーザーが追加操作をキャンセルできるため、ユーザー対話式のシナリオ （データグリッドに新しい項目の追加するなど）で特に便利です。また、追加操作がコミットされるまで、新しい項目がソートされたり、 フィルタ処理でビューから除外されないようにします。

継承元 **CollectionView**
戻り値 **any**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで更新を中断します。

継承元 **CollectionView**
戻り値 **void**

cancelEdit

```
cancelEdit(): void
```

現在の編集トランザクションを終了し、可能であれば項目を元の値に戻します。

継承元 **CollectionView**
戻り値 **void**

cancelNew

```
cancelNew(): void
```

現在の追加トランザクションを終了し、追加前の新しい項目を破棄します。

継承元 **CollectionView**
戻り値 **void**

clearChanges

```
clearChanges(): void
```

itemsAdded 、**itemsRemoved** 、**itemsEdited** の各コレクションの全項目をクリアすることによってすべての変更をクリアします。

このメソッドは、変更をサーバーに確定した後またはデータをサーバーから更新した後に呼び出します。

継承元 **CollectionView**
戻り値 **void**

commitEdit

```
commitEdit(): void
```

現在の編集トランザクションを終了し、適用前の変更を保存します。

継承元 **CollectionView**
戻り値 **void**

commitNew

```
commitNew(): void
```

現在の追加トランザクションを終了し、追加前の新しい項目を保存します。

継承元 **CollectionView**
戻り値 **void**

contains

```
contains(item: any): boolean
```

指定した項目がこのビューに属するかどうかを示す値を返します。

パラメーター

- **item**: **any**
  調べる項目。

継承元 **CollectionView**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

継承元 **CollectionView**
戻り値 **void**

editItem

```
editItem(item: any): void
```

指定した項目の編集トランザクションを開始します。

パラメーター

- **item**: **any**
  編集する項目。

継承元 **CollectionView**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された更新を再開します。

継承元 **CollectionView**
戻り値 **void**

getAggregate

```
getAggregate(aggType: Aggregate, binding: string, currentPage?: boolean): void
```

このコレクション内の項目の集計値を計算します。

パラメーター

- **aggType**: **Aggregate**
  計算する集計のタイプ。
- **binding**: **string**
  集計するプロパティ。
- **currentPage**: **boolean** OPTIONAL
  現在のページの項目だけを含めるかどうか。

継承元 **CollectionView**
戻り値 **void**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

指定した項目をビューの現在の項目に設定します。

パラメーター

- **item**: **any**
  現在の項目として設定する項目。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

ビューの最初の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

ビューの最後の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

ビューの現在の項目の後の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

ビューの指定したインデックスにある項目を現在の項目として設定します。

パラメーター

- **index**: **number**
  現在の項目として設定する項目のインデックス。

継承元 **CollectionView**
戻り値 **boolean**

moveCurrentToPrevious

```
moveCurrentToPrevious(): boolean
```

ビューの現在の項目の前の項目を現在の項目として設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveToFirstPage

```
moveToFirstPage(): boolean
```

最初のページを現在のページとして設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveToLastPage

```
moveToLastPage(): boolean
```

最後のページを現在のページとして設定します。

継承元 **CollectionView**
戻り値 **boolean**

moveToNextPage

```
moveToNextPage(): boolean
```

現在のページの後のページに移動します。

継承元 **CollectionView**
戻り値 **boolean**

moveToPage

```
moveToPage(index: number): boolean
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先ページのインデックス。

継承元 **CollectionView**
戻り値 **boolean**

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

現在のページの前のページに移動します。

継承元 **CollectionView**
戻り値 **boolean**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

collectionChanged イベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  変更の記述が含まれます。

継承元 **CollectionView**
戻り値 **void**

onCurrentChanged

```
onCurrentChanged(e?: EventArgs): void
```

currentChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **CollectionView**
戻り値 **void**

onCurrentChanging

```
onCurrentChanging(e: CancelEventArgs): boolean
```

currentChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs**。

継承元 **CollectionView**
戻り値 **boolean**

onPageChanged

```
onPageChanged(e?: EventArgs): void
```

pageChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **CollectionView**
戻り値 **void**

onPageChanging

```
onPageChanging(e: PageChangingEventArgs): boolean
```

pageChanging イベントを発生させます。

パラメーター

- **e**: **PageChangingEventArgs**
  イベントデータを含む**PageChangingEventArgs**。

継承元 **CollectionView**
戻り値 **boolean**

onSourceCollectionChanged

```
onSourceCollectionChanged(e?: EventArgs): void
```

sourceCollectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **CollectionView**
戻り値 **void**

onSourceCollectionChanging

```
onSourceCollectionChanging(e: CancelEventArgs): boolean
```

sourceCollectionChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs**。

継承元 **CollectionView**
戻り値 **boolean**

refresh

```
refresh(): void
```

現在のソート、フィルタ、およびグループパラメーターを使用してビューを再作成します。

継承元 **CollectionView**
戻り値 **void**

remove

```
remove(item: any): void
```

指定した項目をコレクションから削除します。

パラメーター

- **item**: **any**
  コレクションから削除する項目。

継承元 **CollectionView**
戻り値 **void**

removeAt

```
removeAt(index: number): void
```

指定したインデックスにある項目をコレクションから削除します。

パラメーター

- **index**: **number**
  コレクションから削除する項目のインデックス。 このインデックスは、ソースコレクションに対してではなくビューに対する相対インデックスです。

継承元 **CollectionView**
戻り値 **void**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 **CollectionView**
引数 **NotifyCollectionChangedEventArgs**

currentChanged

現在の項目が変更された後に発生します。

継承元 **CollectionView**
引数 **EventArgs**

currentChanging

現在の項目が変更される前に発生します。

継承元 **CollectionView**
引数 **CancelEventArgs**

pageChanged

ページインデックスが変更された後に発生します。

継承元 **CollectionView**
引数 **EventArgs**

pageChanging

ページインデックスが変更される前に発生します。

継承元 **CollectionView**
引数 **PageChangingEventArgs**

sourceCollectionChanged

sourceCollection プロパティの値が変化した後に発生します。

継承元 **CollectionView**
引数 **EventArgs**

sourceCollectionChanging

sourceCollection プロパティの値が変化する前に発生します。

継承元 **CollectionView**
引数 **CancelEventArgs**

PivotEngine クラス

ファイル `wijmo.olap.js`
モジュール **wijmo.olap**

通常のデータテーブルをOLAPピボットテーブルに対話的に変換するためのユーザーインタフェース を提供します。

itemsSource コレクション内のデータをフィールドのリストに従って表形式にし、 集計されたデータを含む**pivotView** コレクションを作成します。

ピボットテーブルは、データを 1 つ以上のディメンションにグループ化します。これらのディメンションは、 グリッド内の行と列によって表され、データはグリッドセルに格納されます。

コンストラクタ

- constructor

プロパティ

- allowFieldEditing
- async
- autoGenerateFields
- collectionView
- columnFields
- defaultFilterType
- fields
- filterFields
- isServer
- isUpdating
- isViewDefined
- itemsSource
- pivotView
- rowFields
- serverMaxDetail
- serverPollInterval
- serverTimeout
- showColumnTotals
- showRowTotals
- showZeros
- sortableGroups
- sortOnServer
- totalsBeforeData
- valueFields
- viewDefinition

メソッド

- beginUpdate
- cancelPendingUpdates
- deferUpdate
- editField
- endUpdate
- getDetail
- getDetailView
- getKeys

- invalidate
- onError
- onItemsSourceChanged
- onUpdatedView
- onUpdatingView
- onViewDefinitionChanged
- refresh
- removeField

イベント

- error
- itemsSourceChanged
- updatedView
- updatingView
- viewDefinitionChanged

コンストラクタ

constructor

```
constructor(options?: any): PivotEngine
```

PivotEngine クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  フィールドの初期化データを含むJavaScriptオブジェクト。

戻り値 **PivotEngine**

プロパティ

- allowFieldEditing

このPivotEngine が所有するPivotField オブジェクトのプロパティをユーザーが編集できるかどうかを決定する値を取得または設定します。

If you set this property to false, the context menus shown by controls such as the **PivotGrid** and **PivotPanel** will not include an option for changing the field settings.

The default value for this property is **true**.

型 **boolean**

- async

ビューの更新を非同期に行うかどうかを決定する値を取得または設定します。

このプロパティはデフォルトではtrueに設定されます。これにより、大規模なデータセットの集約が 非同期に実行され、UIスレッドの停止が防がれます。

The default value for this property is **true**.

型 **boolean**

autoGenerateFields

エンジンが**itemsSource** に基づいてフィールドを自動的に生成するかどうかを 決定する値を取得または設定します。

このプロパティをtrueに設定すると、エンジンは**itemsSource** 内アイテムの各プロパティに対してフィールドを生成します。自動生成フィールドの**binding** プロパティはプロパティ名に設定され、**header** プロパティはバインディングの最初の文字を大文字にし、各大文字の前にスペースを追加した文字列に設定されます。

たとえば、 'customerName'プロパティは、エンジンに**binding** が 'customerName'に設定され、**header** が 'Customer Name'に設定されたフィールドを作成させます。

文字列を使用してビューリストの1つにフィールドを追加するときは、 **binding** ではなく**header** を指定する必要があります（バインディングとは異なり、フィールドヘッダーは一意でなければなりません）。

The default value for this property is **true**.

型 **boolean**

collectionView

生データを含む**ICollectionView** を取得します。

型 **ICollectionView**

columnFields

出力テーブル内の列として表示されるフィールドを定義する **PivotField** オブジェクトのリストを取得します。

型 **PivotFieldCollection**

defaultFilterType

デフォルトのフィルタタイプを（値または条件に基づいて）取得または設定します。

The default value for this property is **null**, which causes the engine to use **FilterType.Both** on the client or **FilterType.Condition** on the server.

型 **FilterType**

fields

データソースによって公開される**PivotField** オブジェクトのリストを取得します。

このリストは、**itemsSource** プロパティを設定するたびに自動的に作成されます。

ピボットビューは、このリストからビューを定義するリスト （**valueFields** 、**rowFields** 、**columnFields** 、**filterFields** ）にフィールドをコピーすることで定義されます。

たとえば、次のコードは、データソースを**PivotEngine** に割り当てた後、 **rowFields** 、**columnFields** 、**valueFields** の各リストにフィールドを 追加することでビューを定義します。

```
// ピボットエンジンを作成します
var pe = new wijmo.olap.PivotEngine();

// データソースを設定します（フィールドリストを設定します）
pe.itemsSource = this.getRawData();

// Olapビューの構築中は更新を禁止します
pe.beginUpdate();

// 国を行に表示します
pe.rowFields.push('Country');

// カテゴリと製品を列に表示します
pe.columnFields.push('Category');
pe.columnFields.push('Product');

// 売上高の合計をセルに表示します
pe.valueFields.push('Sales');

// ビューの定義を完了します
pe.endUpdate();
```

型 **PivotFieldCollection**

filterFields

フィルタとして使用されるフィールドを定義する**PivotField** オブジェクトのリストを取得します。

このリストのフィールドは出力テーブルには表示されませんが、入力データのフィルタとして使用されます。

型 **PivotFieldCollection**

isServer

エンジンがサーバーの**itemsSource** にバインドされているか、またはローカルデータを使用しているかを決定する値を取得します。

型 **boolean**

isUpdating

エンジンが現在更新されているかどうかを示す値を取得します。

型 **boolean**

isViewDefined

ピボットビューが現在定義されているかどうかを判定する値を取得します。

ピボットビューが定義されているのは、**valueFields** 、**rowFields** 、または**columnFields** リストのいずれかが空でない場合です。

型 **boolean**

itemsSource

分析される生データ、SSASキューブサービスを記述するオブジェクト、またはComponentOne DataEngineサービス用のURLを含む文字列を含む配列または**ICollectionView** を取得または設定します。

1番目のオプション（配列または**ICollectionView** の使用）は最も単純ですが、処理できる生データの量が制限となります。 データセットに約50,000件以上がある場合、配列に生データをロードするには多くの時間がかかります。

このオプションを使用するには、**itemsSource** プロパティを、分析する生データを含む任意のJavaScript配列に設定します。 次に例を示します。

```
var ng = new wijmo.olap.PivotEngine({
    itemsSource = getDataArray(1000);
});
```

2番目のオプション（OLAP SSASキューブへの直接接続）は、**PivotEngine** がSSASサーバーによって提供されるOLAPキューブに直接接続できるようにします。このオプションは、上記のサイズ制限を削除し、数百万または数十億のレコードを持つデータセットを分析できます。

このオプションを使用するには、**itemsSource** プロパティを、コンポーネントがサービスにアクセスする方法を指定する **url** と **cube** の2つのメンバーを持つオブジェクトに設定します。次に例を示します。

```
var ng = new wijmo.olap.PivotEngine({
    itemsSource: {
        url: 'http://ssrs.componentone.com/OLAP/msmdpump.dll',
        cube: 'Adventure Works'
    }
});
```

OLAP SSASキューブに直接接続する場合、ユーザーが値でフィールドをフィルタリングできません。ただ、条件でフィルタリングすることができます。

3番目のオプションであるComponentOne のデータエンジンサービスを使用すると、クライアントに 生データをダウンロードすることなく、サーバーで大きなデータセットを 分析できます。高性能のFlexPivotサービスを使用することも、 MicrosoftのSQL Server Analysis Services OLAPキューブと やり取りすることもできます。

ComponentOneデータエンジンサービスを使用するには、**itemsSource** プロパティをデータサービスのURLを含んだ文字列に設定します。 次に例を示します。

```
var ng = new wijmo.olap.PivotEngine({
    itemsSource: 'http://demos.componentone.com/ASPNET/c1webapi/4.0.20173.114/api/dataengine/cube'
});
```

PivotEngine はビューの定義をサーバーに送信します。サーバーでサマリーが計算され、クライアントに戻されます。

ComponentOne DataEngineデータソースに接続する場合、ユーザーが値でフィールドをフィルタリングできません。ただ、条件でフィルタリングすることができます。

ComponentOne DataEngineサービスの詳細については、

オンラインマニュアルを参照してください。

OlapServerIntroの製品サンプルは、単一の**PivotEngine** で動作するすべてのオプションを示しています。

型 **any**

pivotView

出力ピボットビューを含む**ICollectionView** を取得します。

型 **ICollectionView**

- rowFields

出力テーブル内の行として表示されるフィールドを定義する **PivotField** オブジェクトのリストを取得します。

型 **PivotFieldCollection**

- serverMaxDetail

getDetail メソッドがサーバーから取得するレコードの最大数を取得または設定します。

このプロパティのデフォルト値は**1,000**です。これにより、多くのシナリオで合理的な詳細を提供します。より詳細なレコードを取得できるようにするには、このプロパティの値を増やします。

型 **number**

- serverPollInterval

結果を取得中に進行状況のためにサーバーをポーリングする前にエンジンが待機する時間（ミリ秒単位）を取得または設定します。

このプロパティのデフォルト値は**500**です。これにより、エンジンは0.5秒ごとに状況を更新するためにサーバーをポーリングします。

型 **number**

- serverTimeout

エンジンが結果をサーバーから返すまで待機する最大時間（ミリ秒単位で）を取得または設定します。

このプロパティのデフォルト値は**60,000**ミリ秒や1分です。サーバーの操作が完了するまでにサーバーの処理に時間がかかることが予想される場合は、このプロパティをより高い値に設定します。

型 **number**

- showColumnTotals

出力**pivotView** に、小計または総計を含む 列を含めるかどうかを決定する値を取得または設定します。

The default value for this property is **ShowTotals.GrandTotals**.

型 **ShowTotals**

- showRowTotals

出力**pivotView** に、小計または総計を含む 行を含めるかどうかを決定する値を取得または設定します。

The default value for this property is **ShowTotals.GrandTotals**.

型 **ShowTotals**

- showZeros

Olap出力テーブルで０を使用して欠損値を示すかどうかを決定する値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- sortableGroups

値フィールド（メジャー）をソートするときにエンジンが、グループをソートするか、または各グループ内でのみグループの順序とデータを保持する必要があるかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

- sortOnServer

サーバーから受信したサマリーデータがすでにソートされているかどうかを示す値を取得または設定します。

このプロパティをtrueに設定すると、**PivotEngine** はサーバーから受信したデータをソートしません。

このプロパティは、通常、標準の**PivotEngine** で使用できないカスタムロジックによって適切にソートされた集計データを返すカスタムサーバーと組み合わせて使用する必要があります。

The default value for this property is **false**.

型 **boolean**

- totalsBeforeData

通常のデータ行およびデータ列の前または後に行および列の合計を表示するかどうかを 決定する値を取得または設定します。

この値がtrueに設定されている場合、データ行の上部に合計行が、 通常のデータ列の左側に合計列が表示されます。

The default value for this property is **false**.

型 **boolean**

- valueFields

出力テーブルに集約されるフィールドを定義する**PivotField** オブジェクトのリストを取得します。

型 **PivotFieldCollection**

- viewDefinition

現在のピボットビュー定義をJSON文字列として取得または設定します。

このプロパティは通常、現在のビューをアプリケーション設定として維持するために使用されます。

たとえば、次のコードは、ローカルストレージを使用してビュー定義を 保存およびロードする２つの関数を実装します。

```
// ビューを保存/ロードします
function saveView() {
  localStorage.viewDefinition = pivotEngine.viewDefinition;
}
function loadView() {
  pivotEngine.viewDefinition = localStorage.viewDefinition;
}
```

型 **string**

メソッド

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで、リフレッシュ処理を一時停止します。

戻り値 **void**

cancelPendingUpdates

```
cancelPendingUpdates(): void
```

保留中の非同期のビュー更新をキャンセルします。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/**endUpdate** ブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

戻り値 **void**

editField

```
editField(field: PivotField): void
```

ユーザーがフィールドの設定を編集することができる設定ダイアログを表示します。

パラメーター

- **field**: **PivotField**
  編集する**PivotField** 。

戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって一時停止されたリフレッシュ処理を再開します。

戻り値 **void**

getDetail

```
getDetail(item: any, binding: string): any[]
```

pivotView リスト内のプロパティに基づいて集約されるレコードを含む配列を取得します。

エンジンがPivotEngineサーバーに接続されている場合、返される値は、 非同期に作成される**ObservableArray** です。

パラメーター

- **item**: **any**
  pivotView リスト内のデータ項目。

- **binding**: **string**
  集約されるプロパティの名前。

戻り値 **any**[]

getDetailView

```
getDetailView(item: any, binding: string): ICollectionView
```

pivotView リスト内のプロパティに基づいて集約されるレコードを含む**ICollectionView** を取得します。

パラメーター

- **item**: **any**
  pivotView リスト内のデータ項目。

- **binding**: **string**
  集約されるプロパティの名前。

戻り値 **ICollectionView**

getKeys

```
getKeys(item: any, binding: string): any
```

pivotView リスト内のプロパティに関する情報を含むオブジェクトを取得します。

返されるオブジェクトには、'rowKey'と'colKey'の2つのプロパティがあります。それぞれに 'fields'と'values'の2つの配列が含まれます。同時に、この情報は **PivotEngine** によって集約される値を一意に識別します。

たとえば、2つの行フィールド'Product'および'Country'のと1つの列フィールド'Active'を 持つピボットビューに対して **getKeys** を呼び出すと、 次のようなオブジェクトが返されます。

```
{
    rowKey: {
        fields: [ 'Product', 'Country'],
        values: [ 'Aoba', 'Japan' ]
    },
    colKey: {
        fields: [ 'Active' ],
        values: [ true ]
    }
}
```

このオブジェクトは、1つの集約値の取得に使用されるデータのサブセットを識別します。 この場合、この値は、Active状態がtrueに設定されている、日本で販売された製品 'Aoba'のすべてのデータ項目を表します。

パラメーター

- **item**: **any**
  pivotView リスト内のデータ項目。

- **binding**: **string**
  集約されるプロパティの名前。

戻り値 **any**

invalidate

```
invalidate(): void
```

ビューを無効にして非同期リフレッシュを行います。

戻り値 **void**

onError

```
onError(e: RequestErrorEventArgs): boolean
```

error イベントを発生させます。

パラメーター

- **e**: **RequestErrorEventArgs**
  エラーに関する情報を含む**RequestErrorEventArgs** 。

戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onUpdatingView

```
onUpdatingView(e: ProgressEventArgs): void
```

updatingView イベントを発生させます。

パラメーター

- **e**: **ProgressEventArgs**
  イベントデータを提供する**ProgressEventArgs** 。

戻り値 **void**

onViewDefinitionChanged

```
onViewDefinitionChanged(e?: EventArgs): void
```

viewDefinitionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(force?: boolean): void
```

データを集約し、出力**pivotView** を更新します。

パラメーター

- **force**: **boolean** OPTIONAL
  更新中であってもリフレッシュを行います（**beginUpdate** を参照）。

戻り値 **void**

removeField

```
removeField(field: PivotField): void
```

現在のビューからフィールドを削除します。

パラメーター

- **field**: **PivotField**
  削除する**PivotField** 。

戻り値 **void**

イベント

error

サーバーからデータを取得するときにエラーがあると発生します。

引数 **RequestErrorEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化した後に発生します。

引数 **EventArgs**

updatedView

エンジンが**pivotView** リストの更新を終了した後に発生します。

引数 **EventArgs**

updatingView

エンジンが**pivotView** リストの更新を開始したときに発生します。

引数 **ProgressEventArgs**

viewDefinitionChanged

ビューの定義が変更された後に発生します。

引数 **EventArgs**

PivotField クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**
派生クラス **CubePivotField**

wijmo.olapデータソース内の項目のプロパティを表します。

コンストラクタ

- constructor

プロパティ

- aggregate
- binding
- collectionView
- dataType
- descending
- engine
- filter
- format
- getValue
- header
- isActive
- isContentHtml
- isMeasure
- key
- parentField
- showAs
- sortComparer
- subFields
- visible
- weightField
- width
- wordWrap

メソッド

- onPropertyChanged

イベント

- propertyChanged

コンストラクタ

constructor

```
constructor(engine: PivotEngine, binding: string, header?: string, options?: any): PivotField
```

PivotField クラスの新しいインスタンスを初期化します。

パラメーター

- **engine**: **PivotEngine**
  このフィールドを所有する**PivotEngine** 。
- **binding**: **string**
  このフィールドの連結先のプロパティ。
- **header**: **string** OPTIONAL
  このフィールドを識別するために表示されるヘッダー（デフォルトは連結先と同じ）。
- **options**: **any** OPTIONAL
  フィールドの初期化データを含むJavaScriptオブジェクト。

戻り値 **PivotField**

プロパティ

aggregate

Gets or sets how the field should be summarized.

The default value for this property is **Aggregate.Sum** for numeric fields, and **Aggregate.Count** for other field types.

型 **Aggregate**

binding

フィールドの連結先のプロパティの名前を取得または設定します。

型 **string**

collectionView

このフィールドに連結された**ICollectionView** を取得します。

型 **ICollectionView**

dataType

フィールドのデータ型を取得または設定します。

型 **DataType**

descending

このフィールドのキーを降順でソートするかどうかを決定する値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- engine

このPivotField を所有するPivotEngine への参照を取得します。

型 **PivotEngine**

- filter

このフィールドの値をフィルタ処理するために使用されるPivotFilter への参照を取得します。

キューブデータソースでのメジャーフィールドの場合、フィルターは集計セル値に適用されます。ノンキューブデータソースでのメジャーフィールドの場合、フィルターは生データに適用されます。

型 **PivotFilter**

- format

フィールド値の表示に使用する書式を取得または設定します。

The default value for this property is **'d'** for date fields, **'n0'** for numeric fields, and the empty string for other field types.

型 **string**

- getValue

Gets or sets a function to be used for retrieving the field values.

This property is set to null by default, causing the engine to use the field's **binding** property to retrieve the value.

If specified, the function should take a single parameter that represents the data item being evaluated and should return the calculated value for the item.

型 **Function**

- header

ユーザーインタフェースでこのフィールドを表すために使用される文字列を取得または設定します。

The default value for this property is a capitalized version of the **binding** value.

型 **string**

- isActive

このフィールドが現在ビューで使用されているかどうかを判定する値を 取得または設定します。

このプロパティをtrueに設定すると、フィールドのデータ型に応じて ビューの**rowFields** または**valueFields** に フィールドが追加されます。

型 **boolean**

- isContentHtml

このフィールドの項目がプレーンテキストではなく、HTMLコンテンツを 含むかどうかを示す値を取得または設定します。

The default value for this property is **false**.

型 **boolean**

- isMeasure

フィールドがメジャーかディメンションかを示す値を取得します。

メジャーは「ファクト」とも呼ばれます。これらの値が、通常、フィールドに関する情報を伝達するために集計できる数値です。

ディメンションは、通常、メジャーをカテゴリに分割するために使用できる文字列、日付、またはブール値です。

型 **boolean**

- key

このPivotField のキーを取得します。

通常のフィールドの場合、キーはフィールドのheader です。 CubePivotField インスタンスの場合、キーは フィールドのbinding です。

型 **string**

- parentField

フィールドの親フィールドを取得します。

同じフィールドを［値］リストに複数回ドラッグすると、 フィールドのコピーが作成され、同じ連結を異なるパラメータで 使用できます。コピーは、その親フィールドへの参照を保持します。

型 **PivotField**

- showAs

Gets or sets a value that specifies how to display the aggregate value.

Options for this property are defined by the **ShowAs** enumeration and include differences between the value and the one in the previous row or column, percentages over the row, column, or grand total, and running totals.

This property is similar to the **Show Values As** feature in Excel.

The default value for this property is **ShowAs.NoCalculation**.

型 **ShowAs**

sortComparer

ソート時に値の比較に使用する関数を取得または設定します。

指定された場合、ソート比較関数は、パラメータとして任意の型の値を 2つ取り、最初の値が2番目の値と比べて小さい、等しい、または大きい のいずれであるかを示す値-1、0、または+1を返します。ソート比較関数がnullを返す場合は、標準の組み込み 比較子が使用されます。

この**sortComparer** プロパティを使用すると、カスタム比較アルゴリズムを 提供でき、単純な文字列比較より、ユーザーが期待する結果によく 一致するソートシーケンスが得られる場合があります。

次の例は、**sortComparer** プロパティの一般的な使用方法を示します。

```
/// 製品のリストを定義します
app.products = 'Wijmo,Aoba,Olap,Xuni'.split(',');

// 'app.products'配列の位置で商品をソートします
ng.viewDefinitionChanged.addHandler(function () {
  var fld = ng.fields.getField('Product');
  if (fld) {
    fld.sortComparer = function (val1, val2) {
      return app.products.indexOf(val1) - app.products.indexOf(val2);
    }
  }
});
```

型 **Function**

subFields

このフィールドの子フィールドを取得します。

型 **PivotField**[]

visible

Gets or sets a value indicating whether this field should be displayed in instances of the **PivotPanel** control.

The default value for this property is **true**.

Setting this property to false hides the field any **PivotPanel** controls, preventing users from adding, removing, or changing the the field position in the engine's view definition.

型 **boolean**

weightField

このフィールドの集計の計算時に重みとして使用する**PivotField** を 取得または設定します。

このプロパティがnullに設定されている場合は、すべての値が重み1と見なされます。

このプロパティを使用すると、加重平均や加重合計を計算できます。 たとえば、データに'Quantity'フィールドと'Price'フィールドがある場合に、'Price' フィールドを値フィールドとして使用し、'Quantity'フィールドを 重みとして使用できます。出力には、データの加重平均が含まれます。

型 **PivotField**

width

このフィールドを**PivotGrid** のようなユーザーインタフェースコントロールで表示するために使用する適切な幅を取得または設定します。

型 **number**

wordWrap

このフィールドのコンテンツをセル内で折り返すことができるかどうかを示す値を 取得または設定します。

The default value for this property is **false**.

型 **boolean**

メソッド

onPropertyChanged

```
onPropertyChanged(e: PropertyChangedEventArgs): void
```

propertyChanged イベントを発生させます。

パラメーター

- **e**: **PropertyChangedEventArgs**
  プロパティ名および新旧の値を含む**PropertyChangedEventArgs** 。

戻り値 **void**

イベント

propertyChanged

この**Range** のプロパティ値が変更されると発生します。

引数 **PropertyChangedEventArgs**

PivotFieldCollection クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**
基本クラス **ObservableArray**
表示 ☐継承されたメンバー ☐イベント発生元

PivotField オブジェクトのコレクションを表します。

コンストラクタ

- constructor

プロパティ

- engine
- isUpdating
- maxItems

メソッド

- beginUpdate
- clear
- deferUpdate
- endUpdate
- getField
- implementsInterface
- indexOf
- insert
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

イベント

- collectionChanged

コンストラクタ

constructor

```
constructor(engine: PivotEngine): PivotFieldCollection
```

PivotFieldCollection クラスの新しいインスタンスを初期化します。

パラメーター

- **engine**: **PivotEngine**
  この**PivotFieldCollection** を所有する**PivotEngine** 。

戻り値 **PivotFieldCollection**

プロパティ

engine

このPivotFieldCollection を所有するPivotEngine への参照を取得します。

型 PivotEngine

isUpdating

通知が現在中断されているかどうかを示す値を取得します（beginUpdate およびendUpdate を参照）。

継承元 ObservableArray
型

maxItems

このコレクションで許容されるフィールドの最大数を取得または設定します。

デフォルトでは、このプロパティはnullに設定されます。この場合、任意の数の項目が許容されます。

型 number

メソッド

beginUpdate

```
beginUpdate(): void
```

次にendUpdate が呼び出されるまで通知を中断します。

継承元 ObservableArray
戻り値 void

clear

```
clear(): void
```

配列からすべての項目を削除します。

継承元 ObservableArray
戻り値 void

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数が完了するまでコレクションは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  更新なしで実行する関数。

継承元 **ObservableArray**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **ObservableArray**
戻り値 **void**

getField

```
getField(key: string): PivotField
```

キーに基づいてフィールドを取得します。

パラメーター

- **key**: **string**
  検索する**key** 。

戻り値 **PivotField**

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

指定したインタフェースがサポートされている場合、trueを返します。

パラメーター

- **interfaceName**: **string**
  調べるインタフェースの名前。

継承元 **ObservableArray**
戻り値 **boolean**

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

配列内で項目を検索します。

パラメーター

- **searchElement**: **any**
  配列内で検索する要素。

- **fromIndex**: **number** OPTIONAL
  検索を開始するインデックス。

継承元 **ObservableArray**
戻り値 **number**

insert

```
insert(index: number, item: any): void
```

配列内の指定した位置に項目を挿入します。

パラメーター

- **index**: **number**
  項目を追加する位置。

- **item**: **any**
  配列に追加する項目。

継承元 **ObservableArray**
戻り値 **void**

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

collectionChanged イベントを発生させます。

パラメーター

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  変更の記述が含まれます。

継承元 **ObservableArray**
戻り値 **void**

push

```
push(...item: any[]): number
```

ヘッダーに基づいてフィールドをプッシュできるようにオーバーライドされます。

パラメーター

- **...item**: **any[]**
  配列に追加する 1 つ以上の**PivotField** オブジェクト。

戻り値 **number**

remove

```
remove(item: any): boolean
```

配列から項目を削除します。

パラメーター

- **item**: **any**
  削除する項目。

継承元 **ObservableArray**
戻り値 **boolean**

removeAt

```
removeAt(index: number): void
```

配列内の指定した位置にある項目を削除します。

パラメーター

- **index**: **number**
  削除する項目の位置。

継承元 **ObservableArray**
戻り値 **void**

setAt

```
setAt(index: number, item: any): void
```

配列内の指定した位置にある項目を設定します。

パラメーター

- **index**: **number**
  項目を設定する位置。

- **item**: **any**
  配列に設定する項目。

継承元 **ObservableArray**
戻り値 **void**

slice

```
slice(begin?: number, end?: number): any[]
```

配列の一部のシャローコピーを作成します。

パラメーター

- **begin**: **number** OPTIONAL
  コピーを開始する位置。

- **end**: **number** OPTIONAL
  コピーを終了する位置。

継承元 **ObservableArray**
戻り値 **any[]**

sort

```
sort(compareFn?: Function): this
```

配列の要素を適切な位置にソートします。

パラメーター

- **compareFn**: **Function** OPTIONAL
  ソート順序を定義する関数。 この関数は2つの引数を受け取り、-1（最初の引数の方が2番目の引数より小さい場合）、+1（大きい場合）、0（等しい場合）のいずれかの値を返す必要があります。 これを省略した場合は、各要素の文字列変換に従って辞書順にソートされます。

継承元 **ObservableArray**
戻り値 **this**

splice

```
splice(index: number, count: number, item?: any): any[]
```

配列からの項目の削除と配列への項目の追加の一方または両方を行います。

パラメーター

- **index**: **number**
  項目を追加または削除する位置。

- **count**: **number**
  配列から削除する項目の数。

- **item**: **any** OPTIONAL
  配列に追加する項目。

継承元 **ObservableArray**
戻り値 **any[]**

イベント

collectionChanged

コレクションが変更されたときに発生します。

継承元 **ObservableArray**
引数 **NotifyCollectionChangedEventArgs**

PivotFieldEditor クラス

ファイル	wijmo.olap.js
モジュール	**wijmo.olap**
基本クラス	**Control**
表示	☐ 継承されたメンバー ☐ イベント発生元

PivotField オブジェクトのエディタ。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- field
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateField

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, options?): PivotFieldEditor
```

PivotFieldEditor クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **PivotFieldEditor**

プロパティ

STATIC controlTemplate

PivotFieldEditor コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

field

編集中の**PivotField** への参照を取得または設定します。

型 **PivotField**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

updateEditor

```
updateEditor(): void
```

現在のフィールドの値を反映するためにエディタを更新します。

戻り値 **void**

updateField

```
updateField(): void
```

現在のエディタの値を反映するためにフィールドを更新します。

戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

PivotFilter クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**

PivotField の値の選別に使用するフィルタを表します。

フィルタには、「値によるフィルタ」と「条件によるフィルタ」の2種類があります。

値フィルタを使用すると、ユーザーが分析に含める特定の値を選択できます。これは、リストから特定の値のチェックをオンにすることによって行われます。

条件フィルタには、演算子および値（例えば、「で始まる」および「s」）によって指定される2つの条件が含まれます。条件には、'and'または'or'演算子を使用して結合できます。

PivotEngine がサーバーベースのデータソースに接続されている場合は、条件フィルターのみを使用できます（値フィルターは自動的に非表示になります）。

コンストラクタ

- constructor

プロパティ

- conditionFilter
- filterType
- isActive
- valueFilter

メソッド

- apply
- clear

コンストラクタ

constructor

```
constructor(field: PivotField): PivotFilter
```

PivotFilter クラスの新しいインスタンスを初期化します。

パラメーター
- **field**: **PivotField**
  このフィルタを所有する**PivotField** 。

戻り値 **PivotFilter**

プロパティ

- conditionFilter

このPivotFilter 内のConditionFilter を取得します。

型 **ConditionFilter**

filterType

このフィルタから提供されるフィルタ処理のタイプを取得または設定します。

このプロパティをnullに設定すると、フィルタは、オーナーフィルタの **defaultFilterType** プロパティで定義された値を使用します。

型 **FilterType**

isActive

フィルタがアクティブかどうかを示す値を取得します。

型 **boolean**

valueFilter

この**PivotFilter** 内の**ValueFilter** を取得します。

型 **ValueFilter**

メソッド

apply

```
apply(value): boolean
```

値がフィルタに合致するかどうかを示す値を取得します。

パラメーター

- **value**:
  テストする値。

戻り値 **boolean**

clear

```
clear(): void
```

フィルタをクリアします。

戻り値 **void**

PivotFilterEditor クラス

ファイル	wijmo.olap.js
モジュール	**wijmo.olap**
基本クラス	**Control**
表示	☐ 継承されたメンバー ☐ イベント発生元

PivotFilter オブジェクトのエディタ。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- field
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- clearEditor
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onFinishEditing
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

イベント

- finishEditing
- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, field: PivotField, options?: any): PivotFilterEditor
```

ColumnFilterEditor クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **field**: **PivotField**
  編集する**PivotField** 。
- **options**: **any** OPTIONAL
  エディタの初期化データを含むJavaScriptオブジェクト。

戻り値 **PivotFilterEditor**

プロパティ

STATIC controlTemplate

PivotFilterEditor コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

field

編集中のフィルタを所有する**PivotField** への参照を取得します。

型 **PivotField**

filter

編集中の**PivotFilter** への参照を取得します。

型 **PivotFilter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**`Control`** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**`dispose`** と **`removeEventListener`** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **`Improving scrolling performance with passive listeners`**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 `Control`
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

clearEditor

```
clearEditor(): void
```

変更をフィルタに適用せずに、エディタフィールドをクリアします。

戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能性があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onFinishEditing

```
onFinishEditing(e?: CancelEventArgs): void
```

finishEditing イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs** OPTIONAL

戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

updateEditor

```
updateEditor(): void
```

現在のフィルタ設定を使用してエディタを更新します。

戻り値 **void**

updateFilter

```
updateFilter(): void
```

現在のエディタの値を反映するためにフィルタを更新します。

戻り値 **void**

イベント

finishEditing

ユーザーがフィルタの編集を終了したときに発生します。

引数 **CancelEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

PivotGrid クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**
基本クラス **FlexGrid**
派生クラス **WjPivotGrid**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid コントロールを拡張して、ピボットテーブルを表示します。

このコントロールを使用するには、その**itemsSource** プロパティを **PivotPanel** コントロールのインスタンスまたは**PivotEngine** に設定します。

コンストラクタ

- constructor

プロパティ

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoScroll
- autoSearch
- autoSizeMode
- bottomLeftCells
- cellFactory
- cells
- centerHeadersVertically
- childItemsPath
- clientSize
- cloneFrozenCells
- collapsibleSubtotals
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- customContextMenu
- deferResizing
- detailDialog
- editableCollectionView
- editRange
- engine
- frozenColumns
- frozenRows

- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemValidator
- keyActionEnter
- keyActionTab
- mergeManager
- newRowAtTop
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- rightToLeft
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionMode
- showAlternatingRows
- showColumnFieldHeaders
- showDetailOnDoubleClick
- showDropDown
- showErrors
- showGroups
- showMarquee
- showRowFieldHeaders
- showRowFieldSort
- showSelectedHeaders
- showSort
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- validateEdits
- viewRange
- virtualizationThreshold

メソッド

- addEventListener

- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseColumnsToLevel
- collapseGroupsToLevel
- collapseRowsToLevel
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getDetail
- getDetailView
- getKeys
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow

- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onItemsSourceChanging
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onRefreshed
- onRefreshing
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- showDetail
- startEditing
- toggleDropDownList

イベント

- autoSizedColumn

autoSizedColumn

autoSizedRow

autoSizingColumn

autoSizingRow

beginningEdit

cellEditEnded

cellEditEnding

copied

copying

deletedRow

deletingRow

draggedColumn

draggedRow

draggingColumn

draggingColumnOver

draggingRow

draggingRowOver

formatItem

gotFocus

groupCollapsedChanged

groupCollapsedChanging

itemsSourceChanged

itemsSourceChanging

loadedRows

loadingRows

lostFocus

pasted

pastedCell

pasting

pastingCell

prepareCellForEdit

refreshed

refreshing

resizedColumn

resizedRow

resizingColumn

resizingRow

rowAdded

rowEditEnded

rowEditEnding

rowEditStarted

rowEditStarting

scrollPositionChanged

selectionChanged

selectionChanging

sortedColumn

sortingColumn

updatedLayout

updatedView

- updatingLayout
- updatingView

コンストラクタ

constructor

```
constructor(element: any, options?): PivotGrid
```

PivotGrid クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **PivotGrid**

プロパティ

activeEditor

現在アクティブになっているセルエディタを表す**HTMLInputElement**を取得します。

継承元 **FlexGrid**
型 **HTMLInputElement**

allowAddNew

ユーザーがソースコレクションに項目を追加できるようにグリッドに新規行テンプレートを表示するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、新規行テンプレートは表示されません。

継承元 **FlexGrid**
型 **boolean**

allowDelete

ユーザーが［Delete］キーを押したときにグリッドの選択されている行を削除するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、選択されている行は削除されません。

継承元 **FlexGrid**
型 **boolean**

allowDragging

ユーザーがマウスを使用して行、列、またはその両方をドラッグできるかどうかを決定する値を取得または設定します。

autoScroll プロパティがtrueに設定されている場合、ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされます。

グリッドでは、列のドラッグがデフォルトで有効になっています。

グリッドが連結モードでは、行をドラッグするには特別な 考慮が必要になります。

グリッドが連結モードでは、行をドラッグするとデータソースとの同期が失われます（たとえば行4は6行として参照されることがあります）。 これを回避するには、**draggedRow** イベントを処理し、データを新しい行の位置と同期させる必要があります。

また、**allowSorting** プロパティをfalseに設定する必要があります。そうしないと、行の順序がデータによって決定され、行をドラッグするのは無意味になります。

次のフィドルでは、連結グリッドでの行のドラッグを実行しています。 **Bound Row Dragging (http://jsfiddle.net/Wijmo5/kyg0qsda/)**.

継承元 **FlexGrid**
型 **AllowDragging**

allowMerging

グリッドのどの部分のセル結合を許可するかを取得または設定します。

継承元 **FlexGrid**
型 **AllowMerging**

allowResizing

ユーザーがマウスを使用して行、列、またはその両方をサイズ変更できるかどうかを決定する値を取得または設定します。

サイズ変更が可能な場合は、列ヘッダーセルの右端をドラッグして列を、行ヘッダーセルの下端をドラッグして行をサイズ変更できます。

ヘッダーセルの端をダブルクリックして、行および列をコンテンツに合わせて自動的にサイズ変更することもできます。 自動サイズ変更の動作は、**autoSizeMode** プロパティを使用してカスタマイズできます。

継承元 **FlexGrid**
型 **AllowResizing**

allowSorting

ユーザーが列ヘッダーセルをクリックして列をソートできるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

autoClipboard

グリッドがクリップボードショートカットを処理するかどうかを決定する値を取得または設定します。

次のクリップボードショートカットがあります。

［Ctrl］+［C］、［Ctrl］+［Ins］	グリッドの選択範囲をクリップボードにコピーします。
［Ctrl］+［V］、［Shift］+［Ins］	クリップボードのテキストをグリッドの選択範囲に貼り付けます。

クリップボード操作は、表示されている行および列だけが対象となります。

読み取り専用のセルは、貼り付け操作の影響を受けません。

継承元 **FlexGrid**
型 **boolean**

autoGenerateColumns

グリッドが**itemsSource** に基づいて列を自動的に生成するかどうかを 決定する値を取得または設定します。

列の生成は、少なくとも1項目を含む**itemsSource** プロパティに依存します。このデータ項目が検査され、列が作成されて、プリミティブ値（数値、文字列、Boolean、またはDate）を含むプロパティに連結されます。

プロパティをnullに設定すると、適切なタイプを推測する方法がないため、列は生成されません。このような場合は、**autoGenerateColumns** プロパティをfalseに設定し、明示的に列を作成する必要があります。次に例を示します。

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // データ項目にnull値が含まれる場合があります
  columns: [                  // そのため、列を明示的に定義します
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

継承元 **FlexGrid**
型 **boolean**

autoScroll

ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされるかどうかを決定する値を取得または設定します。

行と列のドラッグは、**allowDragging** プロパティによって制御されます。

このプロパティはデフォルトでtrueに設定されます。

継承元 **FlexGrid**
型 **boolean**

autoSearch

ユーザーが読み取り専用セルに入力するに伴ってグリッドでセルを検索するかどうかを決定する値を取得または設定します。

検索が現在選択されている列(編集不可能な場合)で行われます。検索は現在選択されている行から始まり、大文字と小文字を区別されません。

継承元 **FlexGrid**
型 **boolean**

autoSizeMode

行または列のサイズを自動設定するときにどのセルを考慮に入れるかを取得または設定します。

このプロパティは、ユーザーが列ヘッダの端をダブルクリックしたときの動作を制御します。

デフォルトでは、その列のヘッダセルとデータセルの内容に基づいて列の幅が自動的に設定されます。このプロパティを使用すると、ヘッダのみまたはデータのみに基づいて列の幅を設定するように変更できます。

継承元 **FlexGrid**
型 **AutoSizeMode**

bottomLeftCells

左下のセルを含む**GridPanel** を取得します。

bottomLeftCells パネルは、行ヘッダの下、**columnFooters** パネルの左に表示されます。

継承元 **FlexGrid**
型 **GridPanel**

cellFactory

このグリッドのセルを作成および更新する**CellFactory** を取得または設定します。

継承元 **FlexGrid**
型 **CellFactory**

cells

データセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

centerHeadersVertically

ヘッダーセルのコンテンツを垂直方向の中央揃えで配置するかどうかを決定する値を 取得または設定します。

The default value for this property is **true**.

型 **boolean**

childItemsPath

階層グリッドで子の行の生成に使用される 1 つ以上のプロパティの名前を取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `childItemsPath = 'items'` ）。

項目の異なるレベルに異なる名前を持つ子項目がある場合は、このプロパティに、各レベルの子項目を含むプロパティの名前から成る配列を設定します。（たとえば、 `childItemsPath = ['checks','earnings'];` ）。

サンプル

Show me (https://jsfiddle.net/Wijmo5/kk9j93bL)

継承元 **FlexGrid**
型 **any**

clientSize

コントロールのクライアントサイズを取得します（コントロールのサイズからヘッダーとスクロールバーを引いた値）。

継承元 **FlexGrid**
型 **Size**

cloneFrozenCells

スクロール中に知覚されるパフォーマンスを向上させるには、FlexGridがフリーズされたセルを複製し、別の要素に表示するかどうかを決定する値を取得または設定します。

このプロパティのデフォルト値はnullであり、ブラウザーによって一番適当な設定が選択されます。

継承元 **FlexGrid**
型 **boolean**

collapsibleSubtotals

ユーザーがグリッドで行および列の小計グループの折りたたみと展開を行えるかどうかを 決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

collectionView

グリッドデータを含む**ICollectionView** を取得します。

継承元 **FlexGrid**
型 **ICollectionView**

columnFooters

列フッターセルを保持する**GridPanel** を取得します。

columnFooters パネルは、グリッドセルの下、**bottomLeftCells** パネルの右に表示されます。これを使用して、グリッドデータの下にサマリー情報を表示できます。

以下の例では、**columnFooters** パネルに 1行を追加して、**aggregate** プロパティが設定された列に サマリーデータを表示する方法を示します。

```
function addFooterRow(flex) {

  // 集計を表示するGroupRowを作成します
  var row = new wijmo.grid.GroupRow();

  // 列フッターパネルに行を追加します
  flex.columnFooters.rows.push(row);

  // ヘッダーにシグマを表示します
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

継承元 **FlexGrid**
型 **GridPanel**

columnHeaders

列ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

columnLayout

現在の列レイアウトを定義するJSON文字列を取得または設定します。

列レイアウト文字列は、列とそのプロパティを含む配列を表します。これを使用して、ユーザーが定義した列レイアウトをセッションを越えて保持できます。また、ユーザーが列レイアウトを変更できるアプリケーションで元に戻す/やり直し機能を実装することもできます。

データマップはシリアル化できないため、列レイアウト文字列に**dataMap**　プロパティは含まれていません。

継承元 **FlexGrid**
型 **string**

columns

グリッドの列コレクションを取得します。

継承元 **FlexGrid**
型 **ColumnCollection**

controlRect

コントロールの外接矩形（ページ座標単位）を取得します。

継承元 **FlexGrid**
型 **Rect**

STATIC controlTemplate

FlexGrid コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **FlexGrid**
型 **any**

customContextMenu

グリッドがカスタムコンテキストメニューを提供するかどうかを決定する値を取得または設定します。

カスタムコンテキストメニューには、フィールド設定の変更、フィールドの削除、 グリッドセルの詳細レコードの表示などを行うためのコマンドが含まれます。

The default value for this property is **true**.

型 **boolean**

deferResizing

ユーザーがマウスボタンを放すまで、行および列のサイズ変更を遅らせるかどうかを決定する値を取得または設定します。

デフォルトでは、**deferResizing** はfalseに設定されており、ユーザーがマウスをドラッグするに伴って行および列がサイズ変更されます。このプロパティをtrueに設定すると、グリッドにサイズ変更マーカーが表示され、ユーザーがマウスボタンを放したときに初めて行または列のサイズが変更されます。

継承元 **FlexGrid**
型 **boolean**

detailDialog

Gets a reference to the **DetailDialog** used to display the detail records when the user double-clicks a cell.

This property can be used to customize the content of the **DetailDialog**.

See also the **showDetailOnDoubleClick** property and the **showDetail** method.

型 **DetailDialog**

editableCollectionView

グリッドデータを含む**IEditableCollectionView** を取得します。

継承元 **FlexGrid**
型 **IEditableCollectionView**

editRange

現在編集中のセルを識別する**CellRange** を取得します。

継承元 **FlexGrid**
型 **CellRange**

engine

この**PivotGrid** を所有する**PivotEngine** への参照を取得します。

型 **PivotEngine**

frozenColumns

固定された列の数を取得または設定します。

固定された列は水平方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

frozenRows

静止行の数を取得または設定します。

固定された行は垂直方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

groupHeaderFormat

グループヘッダーコンテンツを作成するために使用される書式文字列を取得または設定します。

この文字列は、任意のテキストと次の置換文字列を含むことができます。

- **{name}**：グループ化されるプロパティの名前。
- **{value}**：グループ化されるプロパティの値。
- **{level}**：グループレベル。
- **{count}**：このグループ内にある項目の合計数。

列がグループ化プロパティに連結されている場合は、列ヘッダーを使用して パラメータを置換し、列の書式とデータマップを使用して `{value}` パラメータを計算します。 列がない場合は、代わりにグループ情報が使用されます。

グループプロパティに連結された非表示の列を追加して、グループヘッダーセルの書式設定をカスタマイズすることもできます。

このプロパティのデフォルト値は
`'{name}: <b>{value}</b>({count:n0} items)'` です。 これは、 `'Country: UK (12 items)'` や
`'Country: Japan (8 items)'` のようなグループヘッダーを作成します。

継承元 **FlexGrid**
型 **string**

headersVisibility

行ヘッダおよび列ヘッダが表示されるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

imeEnabled

編集モードでないときに、グリッドがIME（Input Method Editor）をサポートするかどうかを決定する値を取得または設定します。

このプロパティは、日本語、中国語、韓国語など、IMEサポートを必要とする言語のサイト/アプリケーションにのみ関係します。

継承元 **FlexGrid**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してセル値を変更できるかどうかを判定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

このグリッドのセルのカスタマイズに使用されるフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意のセルに任意の内容を追加できます。そのため、グリッドセルの外観と動作を非常に柔軟に制御することが可能です。

この関数は、セルを含む**GridPanel**、セルの行インデックスと列インデックス、セルを表すHTML要素の4つのパラメーターをとります。通常は、関数内でセル要素の**innerHTML**　プロパティを変更するようにします。

例:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == wijmo.grid.CellType.Cell) {

    // セルにスパークラインを描画します。
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

FlexGridはセルをリサイクルするため、**itemFormatter** によってセルのスタイル属性を変更する場合は、 セルの適切でないスタイル属性を事前にリセットする必要があります。例:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // カスタマイズする属性をリセットします。
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // このセルのcolorおよびbackgroundColor属性をカスタマイズします。
  ...
}
```

複数のクライアントがグリッドのレンダリングをカスタマイズできるようにする場合は（たとえば、ディレクティブや再利用可能なライブラリを作成するときなど）、代わりに**formatItem** イベントを使用することを検討してください。このイベントを使用すると、複数のクライアントがそれぞれ独自のハンドラをアタッチできます。

継承元　　**FlexGrid**
型　　**Function**

itemsSource

グリッドに表示される項目を含む配列または**ICollectionView** を取得または設定します。

継承元　　**FlexGrid**
型　　**any**

itemValidator

セルに有効なデータが含まれているかどうかを決定するバリデータ関数を取得または設定します。

指定された場合、バリデータ関数はセルの行インデックスと列インデックスを含む2つのパラメータをとり、エラーの説明を含む文字列を返す必要があります。

このプロパティは、バインドされていないグリッドを処理する場合に特に便利です。バインドされたグリッドは、代わりに**getError** プロパティを使用して検証できます。

この例では、セルのすぐ上のセルと同じデータがセルに含まれないようにする方法を示します。

```
// 上のセルは、現在のセルと同じ値が含まれていないことを確認します
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'これは重複した値です...'
    }
  }
  return null; // エラーなし
}
```

継承元 **FlexGrid**
型 **Function**

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は**MoveDown** となり、コントロールがカーソルを次の行に移動します。 この動作は標準的なExcel式の動作です。

継承元 **FlexGrid**
型 **KeyAction**

keyActionTab

［Tab］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は、**None** となります。これにより、Tabキーが押されたときにブラウザ上で次または前のコントロールを選択できます。これは、ページのアクセシビリティを向上させるために推奨される設定になります。

以前のバージョンでは、デフォルトは**Cycle** に設定されていました。これにより、コントロールがカーソルを、グリッドを横切って下部に移動しました。これは標準的なExcelの動作ですが、アクセシビリティには適していません。

また、**CycleOut** の設定が存在します。これにより、カーソルがセルを通じて移動（**Cycle** ）してから、最後または最初のセルが選択されたときにページの次/前のコントロールに移動します 。

継承元 **FlexGrid**
型 **KeyAction**

mergeManager

セルの結合方法を決定する**MergeManager** オブジェクトを取得または設定します。

継承元 **FlexGrid**
型 **MergeManager**

newRowAtTop

新しい行テンプレートをグリッドの上部に配置するか、下部に配置するかを示す値を取得または設定します。

newRowAtTop プロパティをtrueに設定した場合、新しい行テンプレートを常に表示したままにする場合は、**frozenRows** プロパティを 1に設定します。これにより、新しい行テンプレートは上部に固定され、ビューからスクロールオフされなくなります。

allowAddNew プロパティがtrueに設定されている場合、および**itemsSource** オブジェクトが新しい項目の追加をサポートしている場合にのみ、新しい行テンプレートが表示されます。

継承元 **FlexGrid**
型 **boolean**

preserveOutlineState

データがリフレッシュされたときに、グリッドがノードの展開/折りたたみ状態を保持するかどうかを決定する値を取得または設定します。

preserveOutlineState プロパティの実装は、JavaScriptの**Map** オブジェクトに基づいています。このオブジェクトはIE 9およびIE 10で利用できません。

継承元 **FlexGrid**
型 **boolean**

preserveSelectedState

データがリフレッシュされたときに、グリッドが行の選択状態を保持するかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

quickAutoSize

グリッドが列のサイズを自動調整するときに精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、自動サイズ変更の機能が無効になります。このプロパティをtrueに設定すると、各列の**quickAutoSize** プロパティの値に従って、その機能が有効になります。null（デフォルト値）に設定した場合、カスタムの**itemFormatter** を持たないグリッドの機能、または**itemFormatter** イベントにアタッチされたハンドラの機能が有効になります。

継承元 **FlexGrid**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rowHeaderPath

行ヘッダーセルを作成するために使用されるプロパティの名前を取得または設定します。

行ヘッダーセルは表示されないし、選択することもできません。アクセシビリティツールと組み合わせて使用することを意図しています。

継承元 **FlexGrid**
型 **string**

rowHeaders

行ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

rows

グリッドの行コレクションを取得します。

継承元 **FlexGrid**
型 **RowCollection**

scrollPosition

グリッドのスクロールバーの値を表す**Point** を取得または設定します。

継承元 **FlexGrid**
型 **Point**

scrollSize

グリッド内容のサイズ（ピクセル単位）を取得します。

継承元 **FlexGrid**
型 **Size**

selectedItems

現在選択されているデータ項目を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selectedRows

現在選択されている行を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selection

現在の選択を取得または設定します。

継承元 **FlexGrid**
型 **CellRange**

selectionMode

現在の選択モードを取得または設定します。

継承元 **FlexGrid**
型 **SelectionMode**

showAlternatingRows

グリッドで交互表示行のセルに'wj-alt'クラスを追加するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、CSSを変更しなくても、交互表示行スタイルを無効にできます。

継承元 **FlexGrid**
型 **boolean**

showColumnFieldHeaders

グリッドで、左上のパネルに列フィールドヘッダーを 表示するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

showDetailOnDoubleClick

ユーザーがセルをダブルクリックしたときに、グリッドが詳細レコードを含むポップアップを表示するかどうかを決定する値を取得または設定します。The default value for this property is **true**.

型 **boolean**

showDropDown

グリッドで、**showDropDown** プロパティがtrueに設定されている列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **FlexGrid**
型 **boolean**

showErrors

グリッドで、検証エラーがあるセルとエラーの説明を含むツールチップに'wj-state-invalid' クラスを追加するかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**itemValidator** または**getError** プロパティを使用して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

showGroups

データグループを区切るためにグリッドにグループ行を挿入するかどうかを決定する値を 取得または設定します。

データグループを作成するには、グリッドの **itemsSource** として使用される **ICollectionView** オブジェクトの **groupDescriptions** プロパティを変更します。

継承元 **FlexGrid**
型 **boolean**

showMarquee

現在の選択範囲の周囲にマーキー要素を表示するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

showRowFieldHeaders

グリッドで、左上のパネルに行フィールドヘッダーを 表示するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

showRowFieldSort

グリッドで、行フィールドの列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

通常の列ヘッダーとは異なり、行フィールドは常に昇順または降順でソートされます。このプロパティをtrueに設定した場合は、 すべての行フィールドヘッダーに常にソートアイコンが表示されます。

The default value for this property is **false**.

型 **boolean**

showSelectedHeaders

選択されているヘッダセルを示すためにクラス名を追加するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

showSort

グリッドで、列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

ソートは、グリッドの **itemsSource** として使用される **ICollectionView** オブジェクトの **sortDescriptions** プロパティによって制御されます。

継承元 **FlexGrid**
型 **boolean**

sortRowIndex

ソートインジケーターを表示する列ヘッダパネルの行のインデックスを取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、**columnHeaders** パネルの最後の行がソート行になります。

継承元 **FlexGrid**
型 **number**

stickyHeaders

ユーザーがドキュメントをスクロールしたときに、列ヘッダーを表示したままにするかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

topLeftCells

左上のセル（列ヘッダーの左）を含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

treeIndent

異なるレベルの行グループをオフセットするインデントを取得または設定します。

継承元 **FlexGrid**
型 **number**

validateEdits

検証に失敗した編集をユーザーがコミットしようとしたときに、グリッドを編集モードのままにするかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**getError** メソッドを呼び出して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

viewRange

現在表示されているセルの範囲を取得します。

継承元 **FlexGrid**
型 **CellRange**

virtualizationThreshold

仮想化を有効にするために必要な最小行数、最小列数、またはその両方を取得または設定します。

このプロパティはデフォルトでゼロに設定されます。つまり、仮想化は常に有効ということです。これにより、バインディングパフォーマンスとメモリー必要量が向上しますが、スクロール時のパフォーマンス低下は犠牲になります。

グリッドの行数が少ない場合（約50〜100）、このプロパティを150などのやや高い値に設定することで、スクロールのパフォーマンスを向上させることができます。これにより、仮想化が無効になり連結処理が遅くなりますが、認識されるスクロールのパフォーマンスが向上する可能性があります。たとえば、以下のコードは、データソースに150を超える項目がある場合、グリッドがセルを仮想化します。

```
// 150を超える項目がある場合はグリッドを仮想化します
theGrid.virtualizationThreshold = 150;
```

このプロパティを200より大きい値に設定することは推奨されません。ロード処理の時間が長くなり、グリッドはすべての行のセルを作成しているときに数秒間フリーズするため、ページ上の要素数が多いためブラウザが遅くなります。

行と列に別々の仮想化しきい値を設定する場合は、**virtualizationThreshold** プロパティを2つの数値を含む配列に設定できます。この場合、最初の数値は行のしきい値として使用され、2番目の数値は列のしきい値として使用されます。たとえば、次のコードでは、グリッドは行を仮想化しますが、列を仮想化しません。

```
// 行（しきい値0）は仮想化しますが、列は仮想化しません（しきい値10,000）
theGrid.virtualizationThreshold = [0, 10000];
```

継承元 **FlexGrid**
型 **any**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 `Control`
戻り値 **HTMLElement**

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

列をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **c**: **number**
  サイズ変更する列のインデックス。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 `FlexGrid`
戻り値 **void**

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

列の範囲をその内容がちょうど収まるようにサイズ変更します。

測定される行の範囲は常に、現在表示されているすべての行 + 現在表示されていない最大2,000行です。グリッドに大量のデータが含まれている場合には（たとえば、50,000行など）、すべての行を測定すると非常に時間がかかる可能性があるため、すべての行は測定されません。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstColumn**: **number** OPTIONAL
  サイズ変更する最初の列のインデックス（デフォルトは最初の列）。
- **lastColumn**: **number** OPTIONAL
  サイズ変更する最後の列のインデックス（デフォルトは最後の列）。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

行をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **r**: **number**
  サイズ変更する行のインデックス。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の列であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

行の範囲をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックス。
- **lastRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックスす。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の行であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

canEditCell

```
canEditCell(r: number, c: number): void
```

指定されたセルが編集可能かどうかを示す値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

継承元 **FlexGrid**
戻り値 **void**

collapseColumnsToLevel

```
collapseColumnsToLevel(level: number): void
```

すべての列を指定されたレベルまで折りたたみます。

パラメーター

- **level**: **number**
  表示する最大列レベル。 0は総計だけを表示することを意味します。 1は最上位グループだけを表示することを意味します。 大きなレベルを指定すると、すべての列が展開されます。

戻り値 **void**

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

すべてのグループ行を指定したレベルに折りたたみます。

パラメーター

- **level**: **number**
  表示する最大のグループレベル。

継承元 **FlexGrid**
戻り値 **void**

collapseRowsToLevel

```
collapseRowsToLevel(level: number): void
```

すべての行を指定されたレベルまで折りたたみます。

パラメーター

- **level**: **number**
  表示する最大行レベル。 0は総計だけを表示することを意味します。 1は最上位グループだけを表示することを意味します。 大きなレベルを指定すると、すべての行が展開されます。

戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

継承元 **FlexGrid**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

継承元 **FlexGrid**
戻り値 **boolean**

focus

```
focus(): void
```

グリッド全体をスクロールしてビューに入れることなくグリッドにフォーカスを設定するようにオーバーライドされます。

継承元 **FlexGrid**
戻り値 **void**

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セル要素の範囲（ビューポート座標単位）を取得します。

このメソッドは、**cells** パネル内のセル（スクロール可能なデータセル）の範囲を返します。その他のパネルにあるセルの範囲を取得するには、該当する**GridPanel** オブジェクトの**getCellBoundingRect** メソッドを使用してください。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

継承元 **FlexGrid**
戻り値 **Rect**

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

グリッドのスクロール可能領域内のセルに格納された値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

継承元 **FlexGrid**
戻り値 **any**

getClipString

```
getClipString(rng?: CellRange): string
```

CellRange の内容を、クリップボードへのコピーに適した文字列として取得します。

非表示の行および列はクリップボード文字列に含まれません。

パラメーター

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **string**

getColumn

```
getColumn(name: string): Column
```

名前または連結に基づいて列を取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **string**
  検索する名前または連結。

継承元 **FlexGrid**
戻り値 **Column**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDetail

```
getDetail(row: number, col: number): any[]
```

指定されたグリッドセルに基づいて集約されたレコードを含む配列を取得します。

パラメーター

- **row**: **number**
  セルを含む行のインデックス。

- **col**: **number**
  セルを含む列のインデックス。

戻り値 **any[]**

getDetailView

```
getDetailView(row: number, col: number): ICollectionView
```

特定のグリッドセルに基づいて集約されたレコードを含む**ICollectionView** を取得します。

パラメーター

- **row**: **number**
  セルを含む行のインデックス。

- **col**: **number**
  セルを含む列のインデックス。

戻り値 **ICollectionView**

getKeys

```
getKeys(row: number, col: number): any
```

特定のセルの集約に使用されるフィールドと値に関する情報を 含むオブジェクトを取得します。

詳細については、@PivotEngine.getKeys メソッドを参照してください。

パラメーター

- **row**: **number**
  セルを含む行のインデックス。
- **col**: **number**
  セルを含む列のインデックス。

戻り値 **any**

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

GridPanel 内のセルの結合範囲を示す**CellRange** を取得します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。
- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **clip**: **boolean** OPTIONAL
  結合範囲をグリッドの現在のビュー範囲にクリップするかどうか。

継承元 **FlexGrid**
戻り値 **CellRange**

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

パラメーター

- **r**: **number**
  検査するセルの行インデックス。
- **c**: **number**
  検査するセルの列インデックス。

継承元 **FlexGrid**
戻り値 **SelectedState**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

指定されたポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

次に例を示します。

```
// ユーザーがグリッドをクリックしたときに、ポイントをヒットテストします
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

パラメーター

- **pt**: **any**
  調べる**Point** （ページ座標単位）、MouseEventオブジェクト、またはポイントのX座標。

- **y**: **any** OPTIONAL
  ポイントのY座標（ページ座標単位、最初のパラメーターが数値の場合）。

継承元 **FlexGrid**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

指定したCellRangeがこのグリッドの行および列コレクションに対して有効かどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックする範囲。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

autoSizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

autoSizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

autoSizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

autoSizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

beginningEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

cellEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

cellEditEnding イベントを発生させます。

パラメーター

- **e**: **CellEditEndingEventArgs**
  イベントデータを含む**CellEditEndingEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

copied イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

copying イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

deletedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

deletingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

draggedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

draggedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

draggingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

draggingColumnOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

draggingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

draggingRowOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

groupCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

groupCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onItemsSourceChanging

```
onItemsSourceChanging(e: CancelEventArgs): boolean
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

loadedRows イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

loadingRows イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

pasted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

pastedCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

pasting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

pastingCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

prepareCellForEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

resizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

resizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

resizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

resizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

rowAdded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

rowEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

rowEditEnding イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

rowEditStarted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

rowEditStarting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

scrollPositionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

selectionChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

sortedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

sortingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

updatedLayout イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

updatingLayout イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

updatingView イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

refresh

```
refresh(fullUpdate?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

継承元 **FlexGrid**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean**
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。
- **recycle**: **boolean** OPTIONAL
  既存の要素を再利用するかどうか。
- **state**: **boolean** OPTIONAL
  既存の要素を保持してその状態を更新するかどうか。

継承元 `FlexGrid`
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この`Control` が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 `Control`
戻り値 **number**

scrollIntoView

```
scrollIntoView(r: number, c: number, refresh?: boolean): boolean
```

指定したセルが画面に入るようにグリッドをスクロールします。

負の行と列のインデックスは無視されるので、以下を呼び出すと、

```
grid.scrollIntoView(200, -1);
```

グリッドは200行目を表示するように垂直にスクロールしますが、水平にスクロールしません。

パラメーター

- **r**: **number**
  画面に入るようにスクロールする行のインデックス
- **c**: **number**
  画面に入るようにスクロールする列のインデックス。
- **refresh**: **boolean** OPTIONAL
  スクロールした新しい位置をすぐ表示するためにグリッドを更新する必要があるかどうかを決定するオプションのパラメータ。

継承元 **FlexGrid**
戻り値 **boolean**

select

```
select(rng: any, show?: any): void
```

セル範囲を選択し、オプションでそのセル範囲が画面に入るようにスクロールします。

select メソッドは、**CellRange** 、と新しい選択範囲を画面に入るようにスクロールするかどうかを示すオプションのブール型パラメータを渡すことで呼び出すことができます。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(new CellRange(1, 1), true);

// 選択範囲（1,1）~（2,4）を指定し、画面に入るようにスクロールしません。
grid.select(new CellRange(1, 1, 2, 4), false);
```

select メソッドを呼び出してインデックスまたは選択する行と列を渡すこともできます。この場合、選択範囲が画面に入るようにスクロールします。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(1, 1);
```

パラメーター

- **rng**: **any**
  選択する範囲。
- **show**: **any** OPTIONAL
  新しい選択範囲が画面に入るようにスクロールするかどうか。

継承元 **FlexGrid**
戻り値 **void**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

グリッドのスクロール可能領域内のセルの値を設定します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **value**: **any**
  セルに格納する値。
- **coerce**: **boolean** OPTIONAL
  列のデータ型に合わせて値を自動的に変更するかどうか。
- **invalidate**: **boolean** OPTIONAL
  グリッドを無効にして変更を表示するかどうか。

継承元 **FlexGrid**
戻り値 **boolean**

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

文字列を行および列に解析し、その内容を特定の範囲に適用します。

非表示の行および列はスキップされます。

パラメーター

- **text**: **string**
  グリッドに解析する、タブと改行で区切られたテキスト。
- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **void**

showDetail

```
showDetail(row: number, col: number): void
```

指定されたグリッドセルの詳細を含むダイアログを表示します。

パラメーター

- **row**: **number**
  セルを含む行のインデックス。
- **col**: **number**
  セルを含む列のインデックス。

戻り値 **void**

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

指定されたセルの編集を開始します。

FlexGrid の編集は、Excelの編集によく似ています。 ［F2］キーを押すか、セルをダブルクリックすると、 グリッドが**完全編集** モードになります。 このモードでは、ユーザーが［Enter］、［Tab］、または［Esc］キーを押すか、マウスを使用して選択範囲を移動するまで、セルエディタはアクティブのままになります。完全編集モードでは、カーソルキーを押しても、グリッドの編集モードは終了しません。

テキストをセルに直接入力すると、グリッドは**クイック編集**モードになります。 このモードでは、ユーザーがEnter、Tab、Esc、またはいずれかの矢印キーを押すまで、セルエディタはアクティブなままになります。

通常、完全編集モードは既存の値を変更する際に使用します。 クイック編集モードは新しいデータをすばやく入力する際に使用します。

編集中に［F2］キーを押すことで、完全編集モードとクイック編集モードを切り替えることができます。

パラメーター

- **fullEdit**: **boolean** OPTIONAL
  ユーザーがカーソルキーを押したときも編集モードのままにするかどうか。 デフォルトはtrueです。
- **r**: **number** OPTIONAL
  編集する行のインデックス。 デフォルトは現在選択されている行です。
- **c**: **number** OPTIONAL
  編集する列のインデックス。 デフォルトは現在選択されている列です。
- **focus**: **boolean** OPTIONAL
  編集開始時に、エディタにフォーカスを与えるかどうか。 デフォルトはtrueです。

継承元 **FlexGrid**
戻り値 **boolean**

toggleDropDownList

```
toggleDropDownList(): void
```

現在選択されているセルに関連付けられたドロップダウンリストボックスを切り替えます。

このメソッドでは、セルが編集モードに入ったとき、またはユーザが特定のキーを押したときに、ドロップダウンリストを自動的に表示することができます。

たとえば、このコードでは、グリッドが編集モードに入るたびに、グリッドにドロップダウンリストが表示されます。

```
// グリッドが編集モードに入ると、ドロップダウンリストを表示する。
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

このコードでは、ユーザーがスペースバーを押した場合、グリッドが編集モードに入った後で、ドロップダウンリストが表示されます。

```
// ユーザーがスペースバーを押したときにドロップダウンリストを表示する。
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

継承元 **FlexGrid**
戻り値 **void**

イベント

autoSizedColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizedRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

beginningEdit

セルが編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnded

セル編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnding

セル編集が終了するときに発生します。

このイベントを使用して検証を実行し、無効な編集を防ぐことができます。 たとえば、次のコードは、ユーザーが文字「a」を含まない値を入力できないようにします。このコードは、編集が適用される前に、編集前と編集後の値を取得する方法を示しています。

```
function cellEditEnding (sender, e) {

  // 編集前と編集後の値を取得します
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // newValに「a」が含まれていない場合は、編集をキャンセルします
  e.cancel = newVal.indexOf('a') < 0;
}
```

cancel パラメータをtrueに設定すると、編集された値が無視されて、グリッドでセルの元の値が維持されます。

また、**stayInEditMode** パラメータをtrueに設定すると、グリッドの編集モードが維持され、編集をコミットする前にユーザーが無効な値を修正できます。

継承元 **FlexGrid**
引数 **CellEditEndingEventArgs**

copied

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーした後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

copying

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーするときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラでコピー操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletedRow

ユーザーが［Del］キーを押して行を削除した後に発生します （**allowDelete** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletingRow

ユーザーが［Delete］キーを押して選択されている行を削除するときに発生します（**allowDelete** プロパティを参照）。

イベントハンドラで行の削除をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedColumn

ユーザーが列のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedRow

ユーザーが行のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumn

ユーザーが列のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumnOver

ユーザーが列を別の位置にドラッグするときに発生します。

ハンドラは、このイベントをキャンセルして、ユーザーが特定の位置に列をドロップしないようにすることができます。次に例を示します。

```
// ドラッグされている列を記憶します
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// 'sales'列がインデックス0にドラッグされないようにします
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRow

ユーザーが行のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRowOver

ユーザーが行を別の位置にドラッグするときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

formatItem

セルを表す要素が作成されたときに発生します。

このイベントを使用してセルを表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、グループ行のセルから'wj-wrap'クラスを削除します。

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

継承元 **FlexGrid**
引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

groupCollapsedChanged

グループが展開または折りたたまれた後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

groupCollapsedChanging

グループが展開または折りたたまれる直前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

itemsSourceChanged

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

itemsSourceChanging

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

loadedRows

グリッド行がデータソースの項目に連結された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

loadingRows

グリッド行がデータソースの項目に連結される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

pasted

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastedCell

ユーザーがクリップボードの内容をセルに貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pasting

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた時に発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastingCell

ユーザーがクリップボードの内容をセルに貼り付けるときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

prepareCellForEdit

エディタセルが作成されたとき、それがアクティブになる前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

resizedColumn

ユーザーが列のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizedRow

ユーザーが行のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingColumn

列がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingRow

行がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowAdded

ユーザーが新規行テンプレートを編集して新しい項目を作成したときに発生します（**allowAddNew** プロパティを参照）。

イベントハンドラで新しい項目の内容をカスタマイズしたり、新しい項目の作成をキャンセルしたりできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnded

行編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnding

行編集が終了するとき、変更が確定またはキャンセルされる前に発生します。

このイベントを **rowEditStarted** イベントと組み合わせて使用して、ディープ連結編集を元に戻す操作を実装できます。次に例を示します。

```
// 編集の開始時にディープ連結値を保存します
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.')> -1) { // ディープ連結
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// 編集がキャンセルされたときにディープ連結値を復元します
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // ユーザーによって編集がキャンセルされました
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarted

行が編集モードに入った後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarting

行が編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

scrollPositionChanged

コントロールがスクロールされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

selectionChanged

選択が変更された後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

selectionChanging

選択が変更される前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortedColumn

ユーザーが列ヘッダをクリックしてソートを適用した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortingColumn

ユーザーが列ヘッダをクリックしてソートを適用する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

updatedLayout

グリッドで内部レイアウトが更新された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

updatedView

グリッドが現在のビューを構成する要素の作成/更新を完了したときに発生します。

グリッドのビューは以下のような操作に応じて更新されます。

- グリッドまたはそのデータソースの更新
- 行または列の追加、削除、変更
- グリッドのサイズ変更またはスクロール
- 選択の変更

継承元 **FlexGrid**
引数 **EventArgs**

updatingLayout

グリッドで内部レイアウトが更新される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

updatingView

現在のビューを構成する要素の作成/更新をグリッドが開始したときに発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

PivotPanel クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**
基本クラス **Control**
派生クラス **WjPivotPanel**
表示 ☐継承されたメンバー ☐イベント発生元

通常のデータテーブルをOLAPピボットテーブルに対話的に変換するためのユーザーインタフェース を提供します。

Olapピボットテーブルは、データを 1 つ以上のディメンションにグループ化します。これらのディメンションは、 グリッド内の行と列によって表され、集約データはグリッドセルに格納されます。

itemsSource プロパティを使用してソースデータを設定し、 **pivotView** プロパティを 使用して集約データを含む出力テーブルを取得します。

コンストラクタ

- constructor

プロパティ

- autoGenerateFields
- collectionView
- columnFields
- controlTemplate
- engine
- fields
- filterFields
- hostElement
- isDisabled
- isTouching
- isUpdating
- isViewDefined
- itemsSource
- pivotView
- restrictDragging
- rightToLeft
- rowFields
- showFieldIcons
- valueFields
- viewDefinition

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate

- initialize
- invalidate
- invalidateAll
- onGotFocus
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onUpdatedView
- onUpdatingView
- onViewDefinitionChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- updatedView
- updatingView
- viewDefinitionChanged

コンストラクタ

constructor

```
constructor(element: any, options?): PivotPanel
```

PivotPanel クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **PivotPanel**

プロパティ

- autoGenerateFields

エンジンが**itemsSource** に基づいて**fields** コレクションを自動的に設定するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

collectionView

生データを含む**ICollectionView** を取得します。

型 **ICollectionView**

columnFields

出力テーブル内の列を定義するフィールドのリストを取得します。

型 **PivotFieldCollection**

STATIC controlTemplate

PivotPanel コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

engine

この**PivotPanel** によって制御される**PivotEngine** を取得または設定します。

型 **PivotEngine**

fields

ビューの構築に使用できるフィールドのリストを取得します。

型 **PivotFieldCollection**

filterFields

出力テーブルの生成時に適用されるフィルタを定義するフィールドのリストを取得します。

型 **PivotFieldCollection**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isViewDefined

ピボットビューが現在定義されているかどうかを判定する値を取得します。

ピボットビューが定義されているのは、**valueFields** リストが空でなく、 かつ**rowFields** リストと**columnFields** リストのいずれかが空でない場合です。

型 **boolean**

- itemsSource

生データを含む配列または**ICollectionView** を取得または設定します。

型 **any**

- pivotView

出力ピボットビューを含む**ICollectionView** を取得します。

型 **ICollectionView**

- restrictDragging

パネルがフィールドのタイプに基づいてドラッグ操作を制限するかどうかを決定する値を取得または設定します。

このプロパティがtrueに設定されている場合、ディメンションフィールドを値フィールドリストにドラッグ、またメジャーフィールドを行または列のフィールドリストにドラッグすることができません。

このプロパティがfalseに設定されている場合、すべてのドラッグ操作が可能になります。

このプロパティがnull（デフォルト値）に設定されている場合、すべてのドラッグ操作が通常のデータソースに対して許可され、キューブデータソースでのドラッグが制限されます。

型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rowFields

出力テーブル内の行を定義するフィールドのリストを取得します。

型 **PivotFieldCollection**

showFieldIcons

メインフィールドリストに、フィールドがメジャーフィールドかディメンションフィールドかを示すアイコンを含めるかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

型 **boolean**

valueFields

出力テーブルに表示される値を定義するフィールドのリストを取得します。

型 **PivotFieldCollection**

viewDefinition

現在のピボットビュー定義をJSON文字列として取得または設定します。

このプロパティは通常、現在のビューをアプリケーション設定として維持するために使用されます。

たとえば、次のコードは、ローカルストレージを使用してビュー定義を 保存およびロードする２つの関数を実装します。

```
// ビューを保存/ロードします
function saveView() {
  localStorage.viewDefinition = pivotPanel.viewDefinition;
}
function loadView() {
  pivotPanel.viewDefinition = localStorage.viewDefinition;
}
```

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

onUpdatingView

```
onUpdatingView(e: ProgressEventArgs): void
```

updatingView イベントを発生させます。

パラメーター

- **e**: **ProgressEventArgs**
  イベントデータを提供する**ProgressEventArgs** 。

戻り値 **void**

onViewDefinitionChanged

```
onViewDefinitionChanged(e?: EventArgs): void
```

viewDefinitionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化した後に発生します。

引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

updatedView

エンジンが**pivotView** リストの更新を終了した後に発生します。

引数 **EventArgs**

updatingView

エンジンが**pivotView** リストの更新を開始したときに発生します。

引数 **ProgressEventArgs**

viewDefinitionChanged

ビューの定義が変更された後に発生します。

引数 **EventArgs**

ProgressEventArgs クラス

ファイル wijmo.olap.js
モジュール **wijmo.olap**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

プログレスイベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- empty
- progress

コンストラクタ

constructor

```
constructor(progress: number): ProgressEventArgs
```

ProgressEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **progress**: **number**
  進捗状況を表す０～100までの数値。

戻り値 **ProgressEventArgs**

プロパティ

STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

progress

現在の進捗状況を０～100の数値で取得します。

型 **number**

Slicer クラス

ファイル	`wijmo.olap.js`
モジュール	**wijmo.olap**
基本クラス	**Control**
派生クラス	**WjSlicer**
表示	☐ 継承されたメンバー ☐ イベント発生元

The **Slicer** control provides a quick way to edit filters applied to **PivotField** objects.

It provides buttons the user can click to filter data based on values and indicates the current filtering state, which makes it easy to understand what is shown in filtered **PivotGrid** and **PivotChart** controls.

For example, when the user selects 'Smith' in a 'Salespersons' field, only data that includes 'Smith' in that field will be included in the output summary.

コンストラクタ

- constructor

プロパティ

- controlTemplate
- field
- header
- hostElement
- isDisabled
- isTouching
- isUpdating
- multiSelect
- rightToLeft
- showCheckboxes
- showHeader

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, options?): Slicer
```

Initializes a new instance of the **Slicer** class.

パラメーター

- **element**: **any**
  The DOM element that hosts the control, or a CSS selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

戻り値 **Slicer**

プロパティ

STATIC controlTemplate

Gets or sets the template used to instantiate **Slicer** controls.

型 **any**

field

Gets or sets the **PivotField** being filtered by this **Slicer**.

If the **PivotField** is not included in the current view definition, the **Slicer** will automatically add the field to the engine's **filterFields** collection.

If you want to remove the field from any **PivotPanel** controls so users cannot remove the field from the view definition, set the fields **visible** property to false. The field will remain active, but will not be shown in any **PivotPanel** controls.

型 **PivotField**

header

Gets or sets the header string shown at the top of the **Slicer**.

The default value for this property is null, which causes the **Slicer** to display the **field** header at the top of the **Slicer**.

型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- multiSelect

Gets or sets a value that determines whether users should be allowed to select multiple values from the list.

The default value for this property is **false**.

型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- showCheckboxes

Gets or sets a value indicating whether the control displays checkboxes next to each item.

The default value for this property is **false**.

型 **boolean**

- showHeader

Gets or sets a value indicating whether the control displays the header area with the header string and multi-select/clear buttons.

The default value for this property is **true**.

型 **boolean**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

DimensionType 列挙体

ファイル `wijmo.olap.js`
モジュール **wijmo.olap**

CubePivotField のディメンションタイプを定義します。

メンバー

名前	値	説明
Dimension	0	データの集約に使用されるカテゴリを含むフィールド。
Measure	1	定量的な数値情報を含むフィールド。
Kpi	2	営業成績の評価に使用されるメジャーグループに関連付けられた計算。
NameSet	3	一連のディメンションメンバを返す多次元式（MDX）。
Attribute	4	ディメンションメンバに関する補足情報を提供します。
Folder	5	メジャーの分類とユーザーのブラウズエクスペリエンスの向上に使用されます。
Hierarchy	6	テーブル内の2つ以上の列の関係を定義するメタデータ。
Date	7	分析とレポートに使用される時間ベースの粒度レベルを持つディメンション。
Currency	8	財務レポートに使用される通貨のリストを表す属性を持つディメンション。

LegendVisibility 列挙体

ファイル `wijmo.olap.js`
モジュール **wijmo.olap**

チャートの凡例を表示するかどうかを定義する定数を指定します。

メンバー

名前	値	説明
Always	0	凡例を常に表示します。
Never	1	凡例を表示しません。
Auto	2	チャートに系列が1つ以上の場合のみ凡例を表示します。

PivotChartType 列挙体

ファイル `wijmo.olap.js`
モジュール **wijmo.olap**

チャートタイプを定義する定数を指定します。

メンバー

名前	値	説明
Column	0	縦棒を表示して、カテゴリ間で項目の値を比較できるようにします。
Bar	1	横棒を表示します。
Scatter	2	X座標とY座標を使用して、データに含まれるパターンを表示します。
Line	3	一定期間のトレンドまたはカテゴリ間のトレンドを表示します。
Area	4	線の下の領域が色で塗りつぶされた折れ線グラフを表示します。
Pie	5	円グラフを表示します。

ShowAs 列挙体

ファイル wijmo.olap.js
モジュール **wijmo.olap**

出力ビューのセルに適用される計算を定義する定数を指定します。

メンバー

名前	値	説明
NoCalculation	0	シンプルな集計値を表示します。
DiffRow	1	項目とその前の行の項目との差を表示します。
DiffRowPct	2	項目とその前の行の項目との差を表示します。
DiffCol	3	項目とその前の列の項目との差を表示します。
DiffColPct	4	項目とその前の列の項目との差をパーセント値で表示します。
PctGrand	5	フィールドに総計に対するパーセント値で値を表示します。
PctRow	6	フィールドに行合計に対するパーセント値で値を表示します。
PctCol	7	フィールドに列合計に対するパーセント値で値を表示します。
RunTot	8	現在の合計値を表示します。
RunTotPct	9	現在の合計値をパーセント値で表示します。

ShowTotals 列挙体

ファイル `wijmo.olap.js`
モジュール **wijmo.olap**

出力テーブルに合計を含めるかどうかを定義する定数を指定します。

メンバー

名前	値	説明
None	0	合計を表示しません。
GrandTotals	1	総計を表示します。
Subtotals	2	小計と総計を表示します。

wijmo.viewer モジュール

ファイル `wijmo.viewer.js`
モジュール **wijmo.viewer**

ビューアコントロールに関連する一連のクラス、インタフェース、および関数を定義します。

クラス

- PdfViewer
- QueryLoadingDataEventArgs
- ReportViewer
- RequestEventArgs
- ViewerBase

インターフェイス

- ICatalogItem
- IHttpRequestHandler
- IPromise

列挙体

- CatalogItemType
- MouseMode
- ViewMode
- ZoomMode

PdfViewer クラス

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**
基本クラス **ViewerBase**
派生クラス **WjPdfViewer**
表示 ☐継承されたメンバー ☐イベント発生元

PDFドキュメントを表示するPDFViewerコントロールを定義します。

serviceUrl プロパティは、PDFサービスを提供するC1 Web APIのURLを示します。 PDFサービスは、C1PdfDocumentSourceを使用してPDFドキュメントを処理します。

次に、PDFドキュメントを表示するサンプルを示します。

```
var pdfViewer = new wijmo.viewer.PdfViewer('#pdfViewer');
pdfViewer.serviceUrl= 'http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report';
pdfViewer.filePath= 'PdfsRoot/C1XapOptimizer.pdf';
```

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filePath
- fullScreen
- hostElement
- isDisabled
- isTouching
- isUpdating
- mouseMode
- pageIndex
- requestHeaders
- rightToLeft
- selectMouseMode
- serviceUrl
- thresholdWidth
- viewMode
- zoomFactor
- zoomMode

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate

- initialize
- invalidate
- invalidateAll
- moveToPage
- onBeforeSendRequest
- onFullScreenChanged
- onGotFocus
- onLostFocus
- onMouseModeChanged
- onPageIndexChanged
- onQueryLoadingData
- onRefreshed
- onRefreshing
- onSelectMouseModeChanged
- onViewModeChanged
- onZoomFactorChanged
- refresh
- refreshAll
- reload
- removeEventListener
- showPageSetupDialog
- zoomToView
- zoomToViewWidth

イベント

- beforeSendRequest
- fullScreenChanged
- gotFocus
- lostFocus
- mouseModeChanged
- pageIndexChanged
- queryLoadingData
- refreshed
- refreshing
- selectMouseModeChanged
- viewModeChanged
- zoomFactorChanged

コンストラクタ

constructor

```
constructor(element: any, options?: any): PdfViewer
```

PdfViewer クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **PdfViewer**

プロパティ

STATIC controlTemplate

ビューアコントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **ViewerBase**
型 **any**

filePath

サーバー上のドキュメントの完全パスを取得または設定します。

パスは、指定されたドキュメントをロードするためにサーバーに登録されているプロバイダのキーで始まります。

継承元 **ViewerBase**
型 **string**

fullScreen

ビューアが全画面表示モードかどうかを示す値を取得または設定します。

継承元 **ViewerBase**
型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

mouseMode

マウスの動作を示す値を取得または設定します。

デフォルトはSelectToolです。その場合は、マウスでクリックしてドラッグすると、テキストが選択されます。

継承元 **ViewerBase**
型 **MouseMode**

pageIndex

ビューパネルに現在表示されているページのインデックスを取得します。

継承元 **ViewerBase**
型 **number**

requestHeaders

データを送信または要求するときに使用する要求ヘッダーを含むオブジェクトを取得または設定します。

このプロパティは、認証が必要なシナリオでよく使用されます。以下に例を示しています。

```
viewer.requestHeaders = {
    Authorization: 'Bearer ' + appAuthService.getToken();
};
```

継承元 **ViewerBase**
型 **any**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectMouseMode

非推奨になりました。代わりにmouseModeを使用してください。

継承元 **ViewerBase**
型 **boolean**

- serviceUrl

C1 Web APIサービスのアドレスを取得または設定します。

例："http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report"。

継承元 **ViewerBase**
型 **string**

- thresholdWidth

モバイルテンプレートとPCテンプレートの切り替えに使用するしきい値を取得または設定します。

デフォルト値は767pxです。コントロールの幅がthresholdWidthより小さい場合は、モバイルテンプレートが適用されます。コントロールの幅がthresholdWidth以上の場合は、PCテンプレートが適用されます。 thresholdWidthを0に設定すると、PCテンプレートだけが適用されます。 9999などの大きな数字に設定すると、モバイルテンプレートだけが適用されます。

継承元 **ViewerBase**
型 **number**

- viewMode

ドキュメントページの表示方法を示す値を取得または設定します。

継承元 **ViewerBase**
型 **ViewMode**

- zoomFactor

ドキュメントページを表示する現在のズーム倍率を示す値を取得または設定します。

継承元 **ViewerBase**
型 **number**

- zoomMode

ドキュメントページを表示する現在のズームモードを示す値を取得または設定します。

継承元 **ViewerBase**
型 **ZoomMode**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

moveToPage

```
moveToPage(index: number): IPromise
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先のページの0から始まるインデックス。

継承元 **ViewerBase**
戻り値 **IPromise**

onBeforeSendRequest

```
onBeforeSendRequest(e: RequestEventArgs): void
```

beforeSendRequest イベントを発生させます。

パラメーター

- **e**: **RequestEventArgs**
  RequestEventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onFullScreenChanged

```
onFullScreenChanged(e?: EventArgs): void
```

fullScreenChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onMouseModeChanged

```
onMouseModeChanged(e?: EventArgs): void
```

mouseModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onPageIndexChanged

```
onPageIndexChanged(e?: EventArgs): void
```

pageIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onQueryLoadingData

```
onQueryLoadingData(e: QueryLoadingDataEventArgs): void
```

queryLoadingData イベントを発生させます。

パラメーター

- **e**: **QueryLoadingDataEventArgs**
  ロード中のデータを含む**QueryLoadingDataEventArgs** オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectMouseModeChanged

```
onSelectMouseModeChanged(e?: EventArgs): void
```

非推奨になりました。代わりにonMouseModeChangedを使用してください。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onViewModeChanged

```
onViewModeChanged(e?: EventArgs): void
```

viewModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onZoomFactorChanged

```
onZoomFactorChanged(e?: EventArgs): void
```

zoomFactorChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **ViewerBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

reload

```
reload(): void
```

ドキュメントを再ロードします。

これは、ドキュメントを強制的に再ロードおよび再レンダリングする場合に便利です。

継承元 **ViewerBase**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

showPageSetupDialog

```
showPageSetupDialog(): void
```

ページ設定ダイアログボックスを表示します。

継承元 **ViewerBase**
戻り値 **void**

zoomToView

```
zoomToView(): void
```

現在のページを拡大縮小して、ビューパネルにページ全体を表示します。

継承元 **ViewerBase**
戻り値 **void**

zoomToViewWidth

```
zoomToViewWidth(): void
```

ビューパネルの幅に合わせて現在のページを拡大縮小します。

継承元 **ViewerBase**
戻り値 **void**

イベント

beforeSendRequest

各要求がサーバーに送信される前に発生します。

このイベントでは、サーバーに送信する前に、URL、ヘッダー、データなどのリクエストオプションに加えてリクエストメソッドも変更できます。 このイベントは、**RequestEventArgs** 型の引数を渡します。そのプロパティは、**httpRequest** メソッドのパラメータと同じ意味と構造を持ち、要求属性を更新するように変更できます。

たとえば、認証トークンを 'Authorization'ヘッダーに入れることができます。

```
viewer.beforeSendRequest.addHandler((s, e) => {
    e.settings.requestHeaders.Authorization = 'Bearer ' + appAuthService.getToken();
});
```

ブラウザで自動的に（たとえば、URLがwindow.open（）関数へのパラメータとして、またはHTMLリンクとして使用される場合）実行されるHTTP要求が誘導するためにURLが使用されている場合、e.settings引数はnullになります。

継承元 **ViewerBase**
引数 **RequestEventArgs**

fullScreenChanged

全画面表示モードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

mouseModeChanged

マウスモードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

pageIndexChanged

ページインデックスが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

queryLoadingData

ドキュメントのロード前に、サービスに送られる要求データをクエリーしたときに発生します。

継承元 **ViewerBase**
引数 **QueryLoadingDataEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectMouseModeChanged

非推奨になりました。代わりにmouseModeChangedを使用してください。

継承元 **ViewerBase**
引数 **EventArgs**

viewModeChanged

ビューモードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

zoomFactorChanged

ズーム倍率が変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

QueryLoadingDataEventArgs クラス

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

queryLoadingData イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- data
- empty

コンストラクタ

constructor

```
constructor(data?: any): QueryLoadingDataEventArgs
```

QueryLoadingDataEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **data**: **any** OPTIONAL
  ドキュメントのロード時に、サービスに送られる要求データ。

戻り値 **QueryLoadingDataEventArgs**

プロパティ

- data

ドキュメントのロード時に、サービスに送られる要求データを取得します。

型 **any**

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

ReportViewer クラス

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**
基本クラス **ViewerBase**
派生クラス **WjReportViewer**
表示 ☐継承されたメンバー ☐イベント発生元

FlexReportまたはSSRSレポートを表示するReportViewerコントロールを定義します。

serviceUrl プロパティは、レポートサービスを提供するC1 Web APIのURLを示します。 レポートサービスは、C1FlexReportを使用してFlexReportを処理し、C1SSRSDocumentSourceとC1PdfDocumentSourceを使用してSSRSレポートを処理します。

次に、FlexReportを表示する方法のサンプルを示します。

```
var reportViewer = new wijmo.viewer.ReportViewer('#reportViewer');
reportViewer.serviceUrl = 'http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report';
reportViewer.filePath = 'ReportsRoot/Formatting/AlternateBackground.flxr';
reportViewer.reportName = 'AlternateBackground';
```

次に、SSRSレポートを表示する方法のサンプルを示します。

```
var reportViewer = new wijmo.viewer.ReportViewer('#reportViewer');
reportViewer.serviceUrl = 'http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report';
reportViewer.filePath = 'c1ssrs/AdventureWorks/Company Sales';
```

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filePath
- fullScreen
- hostElement
- isDisabled
- isTouching
- isUpdating
- mouseMode
- pageIndex
- paginated
- parameters
- reportName
- requestHeaders
- rightToLeft
- selectMouseMode
- serviceUrl
- thresholdWidth
- viewMode
- zoomFactor
- zoomMode

メソッド

- addEventListener
- applyTemplate

- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getReportNames
- getReports
- getTemplate
- initialize
- invalidate
- invalidateAll
- moveToPage
- onBeforeSendRequest
- onFullScreenChanged
- onGotFocus
- onLostFocus
- onMouseModeChanged
- onPageIndexChanged
- onQueryLoadingData
- onRefreshed
- onRefreshing
- onSelectMouseModeChanged
- onViewModeChanged
- onZoomFactorChanged
- refresh
- refreshAll
- reload
- removeEventListener
- showPageSetupDialog
- zoomToView
- zoomToViewWidth

イベント

- beforeSendRequest
- fullScreenChanged
- gotFocus
- lostFocus
- mouseModeChanged
- pageIndexChanged
- queryLoadingData
- refreshed
- refreshing
- selectMouseModeChanged
- viewModeChanged
- zoomFactorChanged

コンストラクタ

constructor

```
constructor(element: any, options?: any): ReportViewer
```

ReportViewer クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **ReportViewer**

プロパティ

STATIC controlTemplate

ビューアコントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **ViewerBase**
型 **any**

filePath

サーバー上のドキュメントの完全パスを取得または設定します。

パスは、指定されたドキュメントをロードするためにサーバーに登録されているプロバイダのキーで始まります。

継承元 **ViewerBase**
型 **string**

fullScreen

ビューアが全画面表示モードかどうかを示す値を取得または設定します。

継承元 **ViewerBase**
型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

mouseMode

マウスの動作を示す値を取得または設定します。

デフォルトはSelectToolです。その場合は、マウスでクリックしてドラッグすると、テキストが選択されます。

継承元 `ViewerBase`
型 `MouseMode`

pageIndex

ビューパネルに現在表示されているページのインデックスを取得します。

継承元 `ViewerBase`
型 **number**

paginated

コンテンツを一連の固定サイズのページとして表すかどうかを示す値を取得または設定します。

デフォルト値はnullです。その場合、FlexReportではページ区切り付きモードが使用され、SSRSレポートではページ区切りなしモードが使用されます。

型 **boolean**

parameters

レポートの実行に使用するパラメータを記述する{name: value}ペアの辞書を取得または設定します。

このプロパティは、レポートで特定のパラメータ（たとえば、非表示のパラメータ）を初期段階で渡す必要がある場合に役に立ちます。

```
reportViewer.parameters = {
   'CustomerID': 'ALFKI'
};
```

型 **any**

reportName

レポート名を取得または設定します。

FlexReportの場合は、FlexReport定義ファイルに定義されたレポート名でこれを設定します。 SSRSレポートの場合は、これを空の文字列のままにしてください。SSRSレポートのパスは、**filePath** プロパティで指定します。

型 **string**

requestHeaders

データを送信または要求するときに使用する要求ヘッダーを含むオブジェクトを取得または設定します。

このプロパティは、認証が必要なシナリオでよく使用されます。以下に例を示しています。

```
viewer.requestHeaders = {
    Authorization: 'Bearer ' + appAuthService.getToken();
};
```

継承元 **ViewerBase**
型 **any**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectMouseMode

非推奨になりました。代わりにmouseModeを使用してください。

継承元 **ViewerBase**
型 **boolean**

serviceUrl

C1 Web APIサービスのアドレスを取得または設定します。

例："http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report"。

継承元 **ViewerBase**
型 **string**

thresholdWidth

モバイルテンプレートとPCテンプレートの切り替えに使用するしきい値を取得または設定します。

デフォルト値は767pxです。コントロールの幅がthresholdWidthより小さい場合は、モバイルテンプレートが適用されます。コントロールの幅がthresholdWidth以上の場合は、PCテンプレートが適用されます。 thresholdWidthを0に設定すると、PCテンプレートだけが適用されます。 9999などの大きな数字に設定すると、モバイルテンプレートだけが適用されます。

継承元 **ViewerBase**
型 **number**

viewMode

ドキュメントページの表示方法を示す値を取得または設定します。

継承元 **ViewerBase**
型 **ViewMode**

zoomFactor

ドキュメントページを表示する現在のズーム倍率を示す値を取得または設定します。

継承元 **ViewerBase**
型 **number**

zoomMode

ドキュメントページを表示する現在のズームモードを示す値を取得または設定します。

継承元 **ViewerBase**
型 **ZoomMode**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

STATIC getReportNames

```
getReportNames(serviceUrl: string, reportFilePath: string, httpHandler?: IHttpRequestHandler): IPromise
```

指定されたFlexReport定義ファイルに定義されたレポート名を取得します。

パラメーター

- **serviceUrl**: **string**
  C1 Web APIサービスのアドレス。
- **reportFilePath**: **string**
  FlexReport定義ファイルの完全パス。
- **httpHandler**: **IHttpRequestHandler** OPTIONAL
  The HTTP request handler. This parameter is optional.

戻り値 **IPromise**

STATIC getReports

```
getReports(serviceUrl: string, path: string, data?: any, httpHandler?: IHttpRequestHandler): IPromise
```

指定されたフォルダパス内のカタログ項目を取得します。

次のデータパラメータを渡すことで、フォルダパスの下のすべての項目を取得できます。 1) true値。 2) true値を含む"recursive"プロパティを持つオブジェクト。

パラメーター

- **serviceUrl**: **string**
  C1 Web APIサービスのアドレス。
- **path**: **string**
  フォルダパス。 FlexReport定義ファイルのパスは、フォルダパスとして扱われます。
- **data**: **any** OPTIONAL
  レポートサービスに送られる要求データ、またはパス下のすべての項目を取得するかどうかを示すboolean値。
- **httpHandler**: **IHttpRequestHandler** OPTIONAL
  The HTTP request handler. This parameter is optional.

戻り値 **IPromise**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

moveToPage

```
moveToPage(index: number): IPromise
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先のページの0から始まるインデックス。

継承元 **ViewerBase**
戻り値 **IPromise**

onBeforeSendRequest

```
onBeforeSendRequest(e: RequestEventArgs): void
```

beforeSendRequest イベントを発生させます。

パラメーター

- **e**: **RequestEventArgs**
  RequestEventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onFullScreenChanged

```
onFullScreenChanged(e?: EventArgs): void
```

fullScreenChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onMouseModeChanged

```
onMouseModeChanged(e?: EventArgs): void
```

mouseModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onPageIndexChanged

```
onPageIndexChanged(e?: EventArgs): void
```

pageIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onQueryLoadingData

```
onQueryLoadingData(e: QueryLoadingDataEventArgs): void
```

queryLoadingData イベントを発生させます。

パラメーター

- **e**: **QueryLoadingDataEventArgs**
  ロード中のデータを含む**QueryLoadingDataEventArgs** オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectMouseModeChanged

```
onSelectMouseModeChanged(e?: EventArgs): void
```

非推奨になりました。代わりにonMouseModeChangedを使用してください。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onViewModeChanged

```
onViewModeChanged(e?: EventArgs): void
```

viewModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onZoomFactorChanged

```
onZoomFactorChanged(e?: EventArgs): void
```

zoomFactorChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **ViewerBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

reload

```
reload(): void
```

ドキュメントを再ロードします。

これは、ドキュメントを強制的に再ロードおよび再レンダリングする場合に便利です。

継承元 **ViewerBase**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

showPageSetupDialog

```
showPageSetupDialog(): void
```

ページ設定ダイアログボックスを表示します。

継承元 **ViewerBase**
戻り値 **void**

zoomToView

```
zoomToView(): void
```

現在のページを拡大縮小して、ビューパネルにページ全体を表示します。

継承元 **ViewerBase**
戻り値 **void**

zoomToViewWidth

```
zoomToViewWidth(): void
```

ビューパネルの幅に合わせて現在のページを拡大縮小します。

継承元 **ViewerBase**
戻り値 **void**

イベント

beforeSendRequest

各要求がサーバーに送信される前に発生します。

このイベントでは、サーバーに送信する前に、URL、ヘッダー、データなどのリクエストオプションに加えてリクエストメソッドも変更できます。 このイベントは、**RequestEventArgs** 型の引数を渡します。そのプロパティは、**httpRequest** メソッドのパラメータと同じ意味と構造を持ち、要求属性を更新するように変更できます。

たとえば、認証トークンを 'Authorization'ヘッダーに入れることができます。

```
viewer.beforeSendRequest.addHandler((s, e) => {
    e.settings.requestHeaders.Authorization = 'Bearer ' + appAuthService.getToken();
});
```

ブラウザで自動的に（たとえば、URLがwindow.open（）関数へのパラメータとして、またはHTMLリンクとして使用される場合）実行されるHTTP要求が誘導するためにURLが使用されている場合、e.settings引数はnullになります。

継承元 **ViewerBase**
引数 **RequestEventArgs**

fullScreenChanged

全画面表示モードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

mouseModeChanged

マウスモードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

pageIndexChanged

ページインデックスが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

queryLoadingData

ドキュメントのロード前に、サービスに送られる要求データをクエリーしたときに発生します。

継承元 **ViewerBase**
引数 **QueryLoadingDataEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectMouseModeChanged

非推奨になりました。代わりにmouseModeChangedを使用してください。

継承元 **ViewerBase**
引数 **EventArgs**

viewModeChanged

ビューモードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

zoomFactorChanged

ズーム倍率が変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

RequestEventArgs クラス

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**
基本クラス **EventArgs**
表示 ☐継承されたメンバー ☐イベント発生元

beforeSendRequest イベントの引数を提供します。

コンストラクタ

- constructor

プロパティ

- empty
- settings
- url

コンストラクタ

constructor

```
constructor(url: string, settings: any): RequestEventArgs
```

RequestEventArgs クラスの新しいインスタンスを初期化します。

パラメーター

- **url**: **string**
  要求が送信されるURLを含む文字列です。
- **settings**: **any**
  リクエストの構築に使用するオブジェクト。 **httpRequest** メソッドの**settings**パラメータと同じ構造とセマンティクスが含まれます。

戻り値 **RequestEventArgs**

プロパティ

- STATIC empty

イベントデータを持たないイベントで使用する値を提供します。

継承元 **EventArgs**
型 **EventArgs**

- settings

要求の構成に使用するオブジェクトを取得または設定します。**httpRequest** メソッドの **settings** パラメータと同じ構造とセマンティクスが含まれます。

型

- url

要求が送信されるURLを取得または設定します。

型

ViewerBase クラス

ファイル	`wijmo.viewer.js`
モジュール	**wijmo.viewer**
基本クラス	**Control**
派生クラス	**PdfViewer**, **ReportViewer**
インターフェイス	**IHttpRequestHandler**
表示	☐ 継承されたメンバー ☐ イベント発生元

すべてのビューアコントロールの基本クラス。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filePath
- fullScreen
- hostElement
- isDisabled
- isTouching
- isUpdating
- mouseMode
- pageIndex
- requestHeaders
- rightToLeft
- selectMouseMode
- serviceUrl
- thresholdWidth
- viewMode
- zoomFactor
- zoomMode

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- moveToPage
- onBeforeSendRequest
- onFullScreenChanged

- onGotFocus
- onLostFocus
- onMouseModeChanged
- onPageIndexChanged
- onQueryLoadingData
- onRefreshed
- onRefreshing
- onSelectMouseModeChanged
- onViewModeChanged
- onZoomFactorChanged
- refresh
- refreshAll
- reload
- removeEventListener
- showPageSetupDialog
- zoomToView
- zoomToViewWidth

イベント

- beforeSendRequest
- fullScreenChanged
- gotFocus
- lostFocus
- mouseModeChanged
- pageIndexChanged
- queryLoadingData
- refreshed
- refreshing
- selectMouseModeChanged
- viewModeChanged
- zoomFactorChanged

コンストラクタ

constructor

```
constructor(element: any, options?: any): ViewerBase
```

ViewerBase クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

戻り値 **ViewerBase**

プロパティ

STATIC controlTemplate

ビューアコントロールのインスタンス化に使用されるテンプレートを取得または設定します。

型 **any**

filePath

サーバー上のドキュメントの完全パスを取得または設定します。

パスは、指定されたドキュメントをロードするためにサーバーに登録されているプロバイダのキーで始まります。

型 **string**

fullScreen

ビューアが全画面表示モードかどうかを示す値を取得または設定します。

型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

mouseMode

マウスの動作を示す値を取得または設定します。

デフォルトはSelectToolです。その場合は、マウスでクリックしてドラッグすると、テキストが選択されます。

型 **MouseMode**

pageIndex

ビューパネルに現在表示されているページのインデックスを取得します。

型 **number**

requestHeaders

データを送信または要求するときに使用する要求ヘッダーを含むオブジェクトを取得または設定します。

このプロパティは、認証が必要なシナリオでよく使用されます。以下に例を示しています。

```
viewer.requestHeaders = {
    Authorization: 'Bearer ' + appAuthService.getToken();
};
```

型 **any**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectMouseMode

非推奨になりました。代わりにmouseModeを使用してください。

型 **boolean**

serviceUrl

C1 Web APIサービスのアドレスを取得または設定します。

例："http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report"。

型 **string**

thresholdWidth

モバイルテンプレートとPCテンプレートの切り替えに使用するしきい値を取得または設定します。

デフォルト値は767pxです。コントロールの幅がthresholdWidthより小さい場合は、モバイルテンプレートが適用されます。コントロールの幅がthresholdWidth以上の場合は、PCテンプレートが適用されます。 thresholdWidthを0に設定すると、PCテンプレートだけが適用されます。 9999などの大きな数字に設定すると、モバイルテンプレートだけが適用されます。

型 **number**

viewMode

ドキュメントページの表示方法を示す値を取得または設定します。

型 **ViewMode**

zoomFactor

ドキュメントページを表示する現在のズーム倍率を示す値を取得または設定します。

型 **number**

zoomMode

ドキュメントページを表示する現在のズームモードを示す値を取得または設定します。

型 **ZoomMode**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

moveToPage

```
moveToPage(index: number): IPromise
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先のページの0から始まるインデックス。

戻り値 **IPromise**

onBeforeSendRequest

```
onBeforeSendRequest(e: RequestEventArgs): void
```

beforeSendRequest イベントを発生させます。

パラメーター

- **e**: **RequestEventArgs**
  RequestEventArgs オブジェクト。

戻り値 **void**

onFullScreenChanged

```
onFullScreenChanged(e?: EventArgs): void
```

fullScreenChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onMouseModeChanged

```
onMouseModeChanged(e?: EventArgs): void
```

mouseModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

戻り値 **void**

onPageIndexChanged

```
onPageIndexChanged(e?: EventArgs): void
```

pageIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

戻り値 **void**

onQueryLoadingData

```
onQueryLoadingData(e: QueryLoadingDataEventArgs): void
```

queryLoadingData イベントを発生させます。

パラメーター

- **e**: **QueryLoadingDataEventArgs**
  ロード中のデータを含む**QueryLoadingDataEventArgs** オブジェクト。

戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectMouseModeChanged

```
onSelectMouseModeChanged(e?: EventArgs): void
```

非推奨になりました。代わりにonMouseModeChangedを使用してください。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

戻り値 **void**

onViewModeChanged

```
onViewModeChanged(e?: EventArgs): void
```

viewModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

戻り値 **void**

onZoomFactorChanged

```
onZoomFactorChanged(e?: EventArgs): void
```

zoomFactorChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

reload

```
reload(): void
```

ドキュメントを再ロードします。

これは、ドキュメントを強制的に再ロードおよび再レンダリングする場合に便利です。

戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

showPageSetupDialog

```
showPageSetupDialog(): void
```

ページ設定ダイアログボックスを表示します。

戻り値 **void**

zoomToView

```
zoomToView(): void
```

現在のページを拡大縮小して、ビューパネルにページ全体を表示します。

戻り値 **void**

zoomToViewWidth

```
zoomToViewWidth(): void
```

ビューパネルの幅に合わせて現在のページを拡大縮小します。

戻り値 **void**

イベント

beforeSendRequest

各要求がサーバーに送信される前に発生します。

このイベントでは、サーバーに送信する前に、URL、ヘッダー、データなどのリクエストオプションに加えてリクエストメソッドも変更できます。 このイベントは、**RequestEventArgs** 型の引数を渡します。そのプロパティは、**httpRequest** メソッドのパラメータと同じ意味と構造を持ち、要求属性を更新するように変更できます。

たとえば、認証トークンを 'Authorization'ヘッダーに入れることができます。

```
viewer.beforeSendRequest.addHandler((s, e) => {
    e.settings.requestHeaders.Authorization = 'Bearer ' + appAuthService.getToken();
});
```

ブラウザで自動的に（たとえば、URLがwindow.open（）関数へのパラメータとして、またはHTMLリンクとして使用される場合）実行されるHTTP要求が誘導するためにURLが使用されている場合、e.settings引数はnullになります。

引数 **RequestEventArgs**

fullScreenChanged

全画面表示モードが変更された後に発生します。

引数 **EventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

mouseModeChanged

マウスモードが変更された後に発生します。

引数 **EventArgs**

pageIndexChanged

ページインデックスが変更された後に発生します。

引数 **EventArgs**

queryLoadingData

ドキュメントのロード前に、サービスに送られる要求データをクエリーしたときに発生します。

引数 **QueryLoadingDataEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectMouseModeChanged

非推奨になりました。代わりにmouseModeChangedを使用してください。

引数 **EventArgs**

viewModeChanged

ビューモードが変更された後に発生します。

引数 **EventArgs**

zoomFactorChanged

ズーム倍率が変更された後に発生します。

引数 **EventArgs**

ICatalogItem インターフェイス

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**

特定のパスのレポートサーバー内の項目を記述します。

プロパティ

- items
- name
- path
- type

プロパティ

- items

子項目の配列。

型 **ICatalogItem**[]

- name

項目の短縮名。

型 **string**

- path

（レポートプロバイダキーから始まる）項目の完全パス。

型 **string**

- type

項目のタイプ。

型 **CatalogItemType**

IHttpRequestHandler インターフェイス

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**

Represents a routine for processing HTTP requests.

プロパティ

- requestHeaders

メソッド

- beforeSend

プロパティ

requestHeaders

Gets or sets an object containing request headers to be used when sending or requesting data.

型 **any**

メソッド

beforeSend

```
beforeSend(args: RequestEventArgs): void
```

Occurs before the request is sent to the server.

パラメーター

- **args**: **RequestEventArgs**
  Describes the current request.

戻り値 **void**

IPromise インターフェイス

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**

非同期呼び出しに使用されるPromiseのインタフェースを定義します。

メソッド

- catch
- then

メソッド

catch

```
catch(onRejected?: (reason?: any)): IPromise
```

Promiseが拒否された後に関数を呼び出します。

パラメーター

- **onRejected**: **(reason?: any)** OPTIONAL
  Promiseが拒否されたときに実行される関数。 これには、1つのパラメータ（拒否理由）があります。 戻り値は、次のコールバック関数に渡されます。

戻り値 **IPromise**

then

```
then(onFulfilled?: (value?: any), onRejected?: (reason?: any)): IPromise
```

Promiseが完了または拒否された後に関数を呼び出します。

パラメーター

- **onFulfilled**: **(value?: any)** OPTIONAL
  Promiseが完了されたときに実行される関数。 これには、1つのパラメータ（完了値）があります。 値が返されると、それが次のコールバック関数に渡されます。 値が返されない場合は、元の値が渡されます。

- **onRejected**: **(reason?: any)** OPTIONAL
  Promiseが拒否されたときに実行される関数。 これには、1つのパラメータ（拒否理由）があります。 値が返されると、それが次のコールバック関数に渡されます。 値が返されない場合は、元の値が渡されます。

戻り値 **IPromise**

CatalogItemType 列挙体

ファイル `wijmo.viewer.js`
モジュール **wijmo.viewer**

カタログ項目のタイプを指定します。

メンバー

名前	値	説明
Folder	0	フォルダ。
File	1	FlexReport定義ファイル。
Report	2	SSRSレポートまたはFlexReport定義ファイルで定義されたFlexReport。

MouseMode 列挙体

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**

マウスモードを指定します。これにより、ビューアのマウス動作を定義できます。

メンバー

名前	値	説明
SelectTool	0	テキストを選択します。
MoveTool	1	ページを移動します。
RubberbandTool	2	選択によるズーム。
MagnifierTool	3	拡大するためのツール。

ViewMode 列挙体

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**

ビューモードを指定します。これは、ビューパネルにドキュメントページを表示する方法を定義します。

メンバー

名前	値	説明
Single	0	1つのドキュメントページだけを表示します。
Continuous	1	ドキュメントページを連続して表示します。

ZoomMode 列挙体

ファイル wijmo.viewer.js
モジュール **wijmo.viewer**

FlexViewerのサポートされるズームモードを記述します。

メンバー

名前	値	説明
Custom	0	カスタムズームモード。 実際のズーム倍率は、 **zoomFactor** プロパティの値によって決定されます。
PageWidth	1	ビューパネル内のページ幅に合うように、必要に応じてページがズームインまたはズームアウトされます。
WholePage	2	ビューパネル内のページ全体に合うように、必要に応じてページがズームインまたはズームアウトされます。

wijmo.angular モジュール

ファイル `wijmo.angular.js`
モジュール **wijmo.angular**

Wijmoコントロール用のAngularJSディレクティブが含まれます。

これらのディレクティブは、HTMLページの単純なマークアップを使用して**AngularJS**アプリケーションにWijmoコントロールを追加できるようにします。

ディレクティブはページマークアップで通常のHTMLタグとして使用できます。タグ名は前に"wj-"が付いたコントロール名に対応し、属性はコントロールのプロパティ名またはイベント名に対応します。

ディレクティブ内のすべてのコントロール名、プロパティ名、およびイベント名は、キャメルケースをハイフンで結んだ小文字の名前に置き換えるという通常のAngularJS命名規則に従います。

AngularJSディレクティブのパラメーターは、使用するバインディングのタイプによって以下の3種類に分けられます。以下の表は、それぞれについて説明します。

`@`	値渡し（一方向バインディング）。属性値はリテラルとして解釈されます。
`=`	参照渡し（双方向バインディング）。属性値は式として解釈されます。
`&`	関数バインディング。属性値は、パラメーターを含む関数呼び出しとして解釈されます。

各バインディングタイプの詳細については、 **ディレクティブに関するDan Wahlinのブログ**を参照してください。 ここではディレクティブイベントについては説明しません。ディレクティブイベントはコントロールイベントと同一であり、バインディングモードは常に同じ（関数バインディング）です。

説明のため、以下に**ComboBox** コントロールを作成するためのマークアップを示します。

```
<wj-combo-box
  text="ctx.theCountry"
  items-source="ctx.countries"
  is-editable="true"
  selected-index-changed="ctx.selChanged(s, e)">
</wj-combo-box>
```

ComboBox の**text**プロパティは、"ctx.theCountry"という名前のコントローラー変数にバインドされています。このバインディングは双方向です。つまり、コントロールの変更によってスコープが更新され、スコープの変更によってコントロールが更新されます。 **text**プロパティを文字列の定数で初期化するには、属性値を一重引用符で囲みます（例: `text="'constant'"` ）。

また、**selected-index-changed**イベントは、"selChanged"という名前のコントローラーメソッドにバインドされており、このバインディングには2つのイベントパラメーターが含まれています（これらのパラメーターがなければ、メソッドは呼び出されません）。 コントロールでこのイベントが発生すると、ディレクティブによってselChangedコントローラーメソッドが呼び出されます。

すべてのWijmo Angular 2コンポーネントには、コントロールがページに追加されて初期化された後に発生する「initialized」イベントがあります。 このイベントを使用して、マークアップでプロパティを設定するほかに、追加的な初期化を実行できます。次に例を示します。次に例を示します。

```
<wj-flex-grid initialized="initGrid(s,e)">
</wj-flex-grid>


// コントローラ
$scope.initGrid: function(s, e) {

  // FlexGridにカスタム結合マネージャーを割り当てます
  s.mergeManager = new CustomMergeManager(s);

}
```

クラス

- WjAutoComplete
- WjBulletGraph
- WjCalendar
- WjCollectionViewNavigator
- WjCollectionViewPager
- WjColorPicker
- WjComboBox

- WjComboBox
- WjContextMenu
- WjFinancialChart
- WjFinancialChartSeries
- WjFlexChart
- WjFlexChartAnimation
- WjFlexChartAnnotation
- WjFlexChartAnnotationLayer
- WjFlexChartAtr
- WjFlexChartAxis
- WjFlexChartBollingerBands
- WjFlexChartBoxWhisker
- WjFlexChartCci
- WjFlexChartChartGestures
- WjFlexChartDataLabel
- WjFlexChartDataPoint
- WjFlexChartEnvelopes
- WjFlexChartErrorBar
- WjFlexChartFibonacci
- WjFlexChartFibonacciArcs
- WjFlexChartFibonacciFans
- WjFlexChartFibonacciTimeZones
- WjFlexChartLegend
- WjFlexChartLineMarker
- WjFlexChartMacd
- WjFlexChartMacdHistogram
- WjFlexChartMovingAverage
- WjFlexChartParametricFunctionSeries
- WjFlexChartRangeSelector
- WjFlexChartRsi
- WjFlexChartSeries
- WjFlexChartStochastic
- WjFlexChartTrendLine
- WjFlexChartWaterfall
- WjFlexChartWilliamsR
- WjFlexChartYFunctionSeries
- WjFlexGrid
- WjFlexGridCellTemplate
- WjFlexGridColumn
- WjFlexGridDetail
- WjFlexGridFilter
- WjFlexPie
- WjFlexPieDataLabel
- WjFlexRadar
- WjFlexRadarAxis
- WjFlexRadarSeries
- WjFlexSheet
- WjGroupPanel
- WjInputColor

- WjInputDate
- WjInputDateTime
- WjInputMask
- WjInputNumber
- WjInputTime
- WjItemTemplate
- WjLinearGauge
- WjListBox
- WjMenu
- WjMenuItem
- WjMenuSeparator
- WjMultiAutoComplete
- WjMultiRow
- WjMultiSelect
- WjPdfViewer
- WjPivotChart
- WjPivotGrid
- WjPivotPanel
- WjPopup
- WjRadialGauge
- WjRange
- WjReportViewer
- WjSheet
- WjSlicer
- WjSunburst
- WjTabPanel
- WjTooltip
- WjTreeMap
- WjTreeView
- WjValidationError

列举体

- CellTemplateType

WjAutoComplete クラス

ファイル	wijmo.angular.js
モジュール	**wijmo.angular**
基本クラス	**WjComboBox**
派生クラス	**WjMultiAutoComplete**
表示	☐継承されたメンバー ☐イベント発生元

AutoComplete コントロール用のAngularJSディレクティブ。

wj-auto-completeディレクティブは、AngularJSアプリケーションに**AutoComplete**コントロールを追加する場合に使用します。ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>AutoCompleteコントロール:</p>
<wj-auto-complete
  text="theCountry"
  items-source="countries"
  is-editable="false"
  placeholder="country">
</wj-auto-complete>
```

次の例では、**AutoComplete**コントロールを作成し、コントローラーによって公開された'countries'配列にコントロールをバインドします。ユーザーが文字を入力すると自動的に国が検索され、候補が絞り込まれて現在の入力と一致するものだけになります。

サンプル

Show me (https://jsfiddle.net/Wijmo5/37GHw)

wj-auto-completeディレクティブは**WjComboBox** を拡張したもので、以下の属性を持ちます。

css-match	@ 検索語に一致する部分のハイライトに使用されるCSSクラスの名前。
delay	@ キーストロークが発生してから検索が実行されるまでの遅延（ミリ秒単位）。
items-source-function	= ユーザーの入力に従って項目を動的に提供する関数。
max-items	@ ドロップダウンに表示する項目の最大数。
min-length	@ オートコンプリート候補を提示するために必要な最小限の入力文字列の長さ。

WjBulletGraph クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
基本クラス **WjLinearGauge**
表示 ☐継承されたメンバー ☐イベント発生元

BulletGraph コントロール用のAngularJSディレクティブ。

wj-bullet-graphディレクティブは、AngularJSアプリケーションにブレットグラフを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<wj-bullet-graph
  value="ctx.gauge.value"
  min="0" max="10"
  target="{{item.target}}"
  bad="{{item.target * .75}}"
  good="{{item.target * 1.25}}">
</wj-bullet-graph>
```

wj-bullet-graphディレクティブは以下の属性をサポートしています。

control	= このディレクティブによって作成されたBulletGraphコントロールへの参照。
direction	@ ゲージの値が増加する方向を示す**GaugeDirection** 値。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
target	@ 指標の目標値。
good	@ 指標が良好と見なされる基準値。
bad	@ 指標が不良と見なされる基準値。
value	= 指標の実際の値。

wj-bullet-graphディレクティブには1つ以上の**WjRange** ディレクティブを含めることができます。

サンプル

Show me (https://jsfiddle.net/Wijmo5/8uxb1vwf)

WjCalendar クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

Calendar コントロール用のAngularJSディレクティブ。

wj-calendarディレクティブは、AngularJSアプリケーションに**Calendar**コントロールを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>Calendarコントロール:</p>
<wj-calendar
  value="theDate">
</wj-calendar>
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/46PhD)

次の例では、**Calendar**コントロールを作成し、コントローラーによって公開された'date'変数にコントロールをバインドします。選択可能な日付の範囲は**min**プロパティと**max**プロパティによって制限されています。

wj-calendarディレクティブは以下の属性をサポートしています。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによってバインドされるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用してスコープにバインドされるコントロールプロパティを指定します。
control	= このディレクティブによって作成された**Calendar** コントロールへの参照。
display-month	= カレンダーに表示する月。
first-day-of-week	@ 週の最初の曜日。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
item-formatter	= カレンダーに表示される日付のカスタマイズに使用される関数。
max	@ 最も遅い有効な日付（書式"yyyy-MM-dd"の文字列）。
min	@ 最も早い有効な日付（書式"yyyy-MM-dd"の文字列）。
month-view	@ コントロールに1か月と1年のどちらを表示するかを示す値。
show-header	@ コントロールにヘッダ領域を表示するかどうかを示す値。
value	= 編集する日付。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。
value-changed	& **valueChanged** イベントハンドラ。

minおよび**max**属性を指定する場合、それらは書式"yyyy-MM-dd"の文字列にする必要があります。技術的には、W3C **[RFC 3339]**で定義された任意のfull-dateを使用できます。これは通常のHTML5入力要素で使用される書式でもあります。

WjCollectionViewNavigator クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐ 継承されたメンバー ☐ イベント発生元

ICollectionView のナビゲーター要素用のAngularJSディレクティブ。

wj-collection-view-navigatorディレクティブを使用すると、**ICollectionView** の項目間を移動するための要素を追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
CollectionViewNavigator:</p>
<wj-collection-view-navigator
  cv="myCollectionView">
</wj-collection-view-navigator>
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/s8tT4)

この例では、100,000件の項目を1ページに20件ずつ表示するCollectionViewを作成します。 現在のページを選択するナビゲーターと現在の項目を選択するナビゲーターが定義されており、データは**FlexGrid** に表示されます。

wj-collection-view-navigatorディレクティブには以下の1つの属性があります。

cv = ナビゲートする**ICollectionView** オブジェクトへの参照。

WjCollectionViewPager クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐ 継承されたメンバー ☐ イベント発生元

ICollectionView のページャー要素用のAngularJSディレクティブ。

wj-collection-view-pagerディレクティブを使用すると、ページ分割された**ICollectionView** のページ間を移動するための要素を追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
CollectionViewPager:</p>
<wj-collection-view-pager
  cv="myCollectionView">
</wj-collection-view-pager>
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/s8tT4)

この例では、100,000件の項目を1ページに20件ずつ表示するCollectionViewを作成します。 現在のページを選択するナビゲーターと現在の項目を選択するナビゲーターが定義されており、データは**FlexGrid** に表示されます。

wj-collection-view-pagerディレクティブには以下の1つの属性があります。

cv = ナビゲートするページ分割された**ICollectionView** オブジェクトへの参照。

WjColorPicker クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

ColorPicker コントロール用のAngularJSディレクティブ。

wj-color-pickerディレクティブは、AngularJSアプリケーションに**ColorPicker**コントロールを追加する場合に使用します。ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>ColorPickerコントロール:</p>
<wj-color-picker
  value="theColor"
  show-alpha-channel="false">
</wj-color-picker>
```

wj-color-pickerディレクティブは以下の属性をサポートしています。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによってバインドされるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用してスコープにバインドされるコントロールプロパティを指定します。
control	= このディレクティブによって作成された**ColorPicker** コントロールへの参照。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
show-alpha-channel	@ コントロールにアルファチャネル（透明度）エディタを表示するかどうかを示す値。
show-color-string	@ コントロールに編集する色の文字列表現を表示するかどうかを示す値。
palette	= パレットとして使用する10個の色の値の配列。
value	= 編集する色。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。
value-changed	& **valueChanged** イベントハンドラ。

WjComboBox クラス

ファイル	wijmo.angular.js
モジュール	**wijmo.angular**
派生クラス	**WjAutoComplete**, **WjInputTime**, **WjMenu**, **WjMultiSelect**
表示	☐継承されたメンバー ☐イベント発生元

ComboBox コントロール用のAngularJSディレクティブ。

wj-combo-boxディレクティブは、AngularJSアプリケーションに**ComboBox**コントロールを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>ComboBoxコントロール:</p>
<wj-combo-box
  text="theCountry"
  items-source="countries"
  is-editable="false"
  placeholder="country">
</wj-combo-box>
```

次の例では、**ComboBox**コントロールを作成し、コントローラーによって公開された'countries'配列にコントロールをバインドします。ユーザーが文字を入力すると、自動的に国が検索されます。**isEditable**プロパティはfalseに設定されているので、ユーザーはリストに存在する項目を選択する必要があります。

サンプル

Show me (https://jsfiddle.net/Wijmo5/37GHw)

wj-combo-boxディレクティブは以下の属性をサポートしています。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**selectedValue**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによってバインドされるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用してスコープにバインドされるコントロールプロパティを指定します。
control	= このディレクティブによって作成された**ComboBox** コントロールへの参照。
display-member-path	@ 項目の視覚表示として使用するプロパティの名前。
is-content-html	@ ドロップダウンリストの項目がプレーンテキストとHTMLのどちらで表示されるかを示す値。
is-dropped-down	@ ドロップダウンが現在表示されているかどうかを示す値。
is-editable	@ ユーザーがリストに存在しない値を入力できるかどうかを示す値。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
item-formatter	= ドロップダウンリストに表示される値のカスタマイズに使用される関数。
items-source	= リストに表示する項目を含む配列または**ICollectionView**。
max-drop-down-height	@ ドロップダウンリストの最大の高さ。
max-drop-down-width	@ ドロップダウンリストの最大の幅。
placeholder	@ コントロールが空のときにヒントとして表示される文字列。
is-required	@ null値が禁止されるかどうかを示す値。
show-drop-down-button	@ コントロールにドロップダウンボタンを表示するかどうかを示す値。
selected-index	= ドロップダウンリストで現在選択されている項目のインデックス。
selected-item	= ドロップダウンリストで現在選択されている項目。
selected-value	= **selected-value-path**を使用して取得された、選択されている項目の値。
selected-value-path	@ **selected-item**から**selected-value**を取得するために使用されるプロパティの名前。
text	= コントロールに表示するテキスト。
is-dropped-down-changing	& **isDroppedDownChanging** イベントハンドラ。
is-dropped-down-changed	& **isDroppedDownChanged** イベントハンドラ。
selected-index-changed	& **selectedIndexChanged** イベントハンドラ。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。
text-changed	& **textChanged** イベントハンドラ。

WjContextMenu クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

コンテキストメニュー用のAngularJSディレクティブ。

wj-context-menuディレクティブは、ページ上の要素にコンテキストメニューを追加する場合に使用します。wj-context-menuディレクティブは**wj-menu**ディレクティブを基にしており、ユーザーが要素のコンテキストメニューを要求したとき（通常は右クリックしたとき）にポップアップメニューを表示します。

wj-context-menuディレクティブは、コンテキストメニューを適用する対象の要素に対する追加パラメーターとして指定します。パラメーターの値は、メニューを含む要素のCSSセレクターです。次に例を示します。

```
<!-- コンテキストメニューのある段落 -->
<p wj-context-menu="#idMenu" >
 この段落にはコンテキストメニューがあります。</p>

<!-- コンテキストメニューを定義します（非表示、ID付き）。 -->
<wj-menu id="idMenu" ng-show="false">
 <wj-menu-item cmd="cmdOpen" cmd-param ="1">開く...</wj-menu-item>
 <wj-menu-item cmd="cmdSave" cmd-param="2">保存</wj-menu-item>
 <wj-menu-item cmd="cmdSave" cmd-param="3">名前を付けて保存...</wj-menu-item>
 <wj-menu-item cmd="cmdNew" cmd-param ="4">新規作成...</wj-menu-item>
 <wj-menu-separator></wj-menu-separator>
 <wj-menu-item cmd="cmdExit" cmd-param="5">終了</wj-menu-item>
</wj-menu >
```

WjFinancialChart クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart コントロール用のAngularJSディレクティブ。

wj-financial-chartディレクティブは、AngularJSアプリケーションに金融チャートを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。

wj-financial-chartディレクティブは以下の属性をサポートしています。

binding @ チャートのYの値を含むプロパティの名前。これは系列レベルでオーバーライドできます。
binding-x @ チャートのXの値を含むプロパティの名前。これは系列レベルでオーバーライドできます。
chart-type @ 系列オブジェクトのレンダリング時に使用するデフォルトのチャートタイプ。これは、系列レベルでオーバーライドできます。**FinancialChartType** を参照してください。
control = このディレクティブによって作成された**FinancialChart** コントロールへの参照。
footer @ チャートのフッタに表示するテキスト（プレーンテキスト）。
footer-style = チャートのフッタに適用するスタイル。
header @ チャートのヘッダに表示するテキスト（プレーンテキスト）。
header-style = チャートのヘッダに適用するスタイル。
initialized & このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized = バインディングによるコントロールの初期化が完了したかどうかを示す値。
interpolate-nulls @ データにnull値が存在するときにギャップを補間するか、そのままにするかを示す値。
item-formatter = データポイントの外観をカスタマイズするフォーマッター関数。
items-source = チャートの作成に使用されるデータを含む配列または**ICollectionView** オブジェクト。
legend-toggle @ 凡例項目をクリックしたときに系列の表示/非表示を切り替えるかどうかを示す値。
options = 特定のチャートタイプにのみ適用するチャートオプション。 詳細については、**FinancialChart** の**options**を参照してください。
palette = 各系列の表示に使用されるデフォルト色を含む配列。
plot-margin = コントロールの端からプロット領域の端までの間隔のピクセル数、またはCSSスタイルのマージン。
selection = 選択されている系列オブジェクト。
selection-mode @ ユーザーが系列をクリックしたときに何が選択されるかを示す**SelectionMode** 値。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。これは系列レベルでオーバーライドできます。
tooltip-content @ **ChartTooltip** に表示する内容。
got-focus & **gotFocus** イベントハンドラ。
lost-focus & **lostFocus** イベントハンドラ。
rendering & **rendering** イベントハンドラ。
rendered & **rendered** イベントハンドラ。
series-visibility-changed & **seriesVisibilityChanged** イベントハンドラ。
selection-changed & **selectionChanged** イベントハンドラ。

wj-financial-chartディレクティブには、子ディレクティブとして **WjFlexChartAxis**、**WjFlexChartSeries**、**WjFlexChartLegend**、および**WjFlexChartDataLabel** を含めることができます。

WjFinancialChartSeries クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart FinancialSeries オブジェクトのAngularJSディレクティブ。

wj-financial-chart-seriesディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

axis-x @ 系列のX軸。
axis-y @ 系列のY軸。
binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
chart-type @ この系列オブジェクトに対応するオブジェクトのレンダリング時に使用するチャートタイプ。この値は、チャートに設定されたデフォルトのチャートタイプをオーバーライドします。**FinancialChartType** を参照してください。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
altStyle = 系列の代替スタイル。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、SplineSymbolsの各チャートでこの系列の データポイントのレンダリングに使用するシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、SplineSymbolsの各チャートでこの系列の データポイントのレンダリングに使用するシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうかと、その場所を示す**SeriesVisibility** 値。

通常、**wj-financial-chart-series**では、**name**プロパティと**binding**プロパティだけを指定します。 残りの値は、親**wj-financial-chart**ディレクティブから継承されます。

WjFlexChart クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐ 継承されたメンバー ☐ イベント発生元

FlexChart コントロール用のAngularJSディレクティブ。

wj-flex-chartディレクティブは、AngularJSアプリケーションにチャートを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>FlexChartコントロール:</p>
<wj-flex-chart
  style="height:300px"
  items-source="data"
  binding-x="country">
  <wj-flex-chart-axis
    wj-property="axisY"
    major-unit="5000">
  </wj-flex-chart-axis>
  <wj-flex-chart-series
    binding="sales"
    name="Sales">
  </wj-flex-chart-series>
  <wj-flex-chart-series
    binding="expenses"
    name="Expenses">
  </wj-flex-chart-series>
  <wj-flex-chart-series
    binding="downloads"
    name="Downloads"
    chart-type="LineSymbols">
  </wj-flex-chart-series>
</wj-flex-chart>
```

次の例では、**FlexChart** コントロールを作成し、コントローラーによって公開された'data'配列にコントロールをバインドします。チャートには3つの系列オブジェクトがあり、各系列がソース配列に含まれるオブジェクトの1つのプロパティに対応します。この例の最後の系列では、'chart-type'属性を使用して、他の系列オブジェクトで使用されているデフォルトのチャートタイプをオーバーライドしています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/QNb9X)

wj-flex-chartディレクティブは以下の属性をサポートしています。

binding @ チャートのYの値を含むプロパティの名前。これは系列レベルでオーバーライドできます。
binding-x @ チャートのXの値を含むプロパティの名前。これは系列レベルでオーバーライドできます。
chart-type @ 系列オブジェクトのレンダリング時に使用するデフォルトのチャートタイプ。これは、系列レベルでオーバーライドできます。**ChartType** を参照してください。
control = このディレクティブによって作成された**FlexChart** コントロールへの参照。
footer @ チャートのフッタに表示するテキスト（プレーンテキスト）。
footer-style = チャートのフッタに適用するスタイル。
header @ チャートのヘッダに表示するテキスト（プレーンテキスト）。
header-style = チャートのヘッダに適用するスタイル。
initialized & このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized = バインディングによるコントロールの初期化が完了したかどうかを示す値。
interpolate-nulls @ データにnull値が存在するときにギャップを補間するか、そのままにするかを示す値。
item-formatter = データポイントの外観をカスタマイズするフォーマッター関数。
items-source = チャートの作成に使用されるデータを含む配列または**ICollectionView** オブジェクト。
legend-toggle @ 凡例項目をクリックしたときに系列の表示/非表示を切り替えるかどうかを示す値。
options = 特定のチャートタイプにのみ適用するチャートオプション。 詳細については、**FlexChart** の**options**を参照してください。
palette = 各系列の表示に使用されるデフォルト色を含む配列。
plot-margin = コントロールの端からプロット領域の端までの間隔のピクセル数、またはCSSスタイルのマージン。
rotated @ X軸が縦、Y軸が横になるように軸を入れ替えるかどうかを示す値。
selection = 選択されている系列オブジェクト。
selection-mode @ ユーザーが系列をクリックしたときに何が選択されるかを示す**SelectionMode** 値。
stacking @ 系列オブジェクトを積み重ねるか個別にプロットするか、積み重ねる場合はどのように積み重ねるかを示す**Stacking** 値。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。これは系列レベルでオーバーライドできます。

tooltip-content	@	**ChartTooltip** に表示する内容。
got-focus	&	**gotFocus** イベントハンドラ。
lost-focus	&	**lostFocus** イベントハンドラ。
rendering	&	**rendering** イベントハンドラ。
rendered	&	**rendered** イベントハンドラ。
series-visibility-changed	&	**seriesVisibilityChanged** イベントハンドラ。
selection-changed	&	**selectionChanged** イベントハンドラ。

wj-flex-chartディレクティブには、子ディレクティブとして **WjFlexChartAxis**、**WjFlexChartSeries**、**WjFlexChartLegend**、および**WjFlexChartDataLabel** を含めることができます。

WjFlexChartAnimation クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart ChartAnimation オブジェクトのAngularJSディレクティブ。

wj-flex-chart-animationディレクティブは、**WjFlexChart**、**WjFlexPie**、または**WjFinancialChart** ディレクティブに含める必要があります。 このディレクティブは以下の属性をサポートしています。

animation-mode @ プロットポイントのアニメーション表示を一度に 1 つずつ行うか、系列ごとに行うか、または一度にすべて行うかを示す値。
easing @ アニメーションに適用されるイージング関数を示す値。
duration @ アニメーション全体の長さ（ミリ秒）を示す値。
axis-animation @ 軸アニメーションが有効かどうかを示す値。

WjFlexChartAnnotation クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

注釈のAngularJSディレクティブ。

wj-flex-chart-annotationディレクティブは、**WjFlexChartAnnotationLayer** ディレクティブに含める必要があります。

wj-flex-chart-annotationディレクティブは、**Circle**、**Rectangle**、**Polygon**など、使用可能なすべてのタイプの注釈の形状を表すために使用されます。注釈の形状のタイプは、ディレクティブの**type**属性で指定されます。

次の属性をサポートします。

type	@ ディレクティブによって表される注釈の形状のクラス名。 有効な値は、**Circle**、**Ellipse**、**Image**、**Line**、**Polygon**、**Rectangle**、**Square**、**Text** です。
attachment	@ 注釈の添付方法を定義する**AnnotationAttachment** 値。
content	@ **Circle**、**Ellipse**、**Image**、**Line**、**Polygon**、**Rectangle**、または**Square**注釈のテキスト。
end	@ **Line**注釈の終点。
height	@ **Ellipse**、**Image**、または**Rectangle**注釈の高さ。
href	@ **Image**注釈のhref。
is-visible	@ 注釈の表示/非表示。
length	@ **Square**注釈の長さ。
name	@ 注釈の名前。
offset	@ 注釈のオフセット。
point	@ 注釈のポイント。ポイントの座標空間は、**attachment**プロパティ値に依存します。 このプロパティは、**Circle**、**Ellipse**、**Image**、**Rectangle**、**Square**、および**Text**注釈に対して機能します。
point-index	@ 注釈がアタッチされる指定された系列内のデータポイントのインデックス。
position	@ **point**に対する注釈の位置を定義する**AnnotationPosition** 値。
radius	@ **Circle**注釈の半径。
series-index	@ 注釈がアタッチされるデータ系列のインデックス。
start	@ **Line**注釈の始点。
style	@ 注釈のスタイル。
text	@ **Text**注釈のテキスト。
tooltip	@ 注釈のツールチップ。
width	@ **Ellipse**、**Image**、または**Rectangle**注釈の幅。

WjFlexChartAnnotationLayer クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart AnnotationLayer オブジェクトのAngularJSディレクティブ。

wj-flex-chart-annotation-layerディレクティブは、**WjFlexChart** ディレクティブまたは**WjFinancialChart** ディレクティブに含める必要があります。

WjFlexChartAtr クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart ATR オブジェクトのAngularJSディレクティブ。

wj-flex-chart-atrディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
period @ ATR（Average True Range）計算の期間。

WjFlexChartAxis クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart Axis オブジェクト用のAngularJSディレクティブ。

wj-flex-chart-axisディレクティブは、**WjFlexChart** ディレクティブまたは**WjFinancialChart** ディレクティブに含まれる必要があります。 このディレクティブは以下の属性をサポートしています。

wj-property @ このディレクティブで初期化する**FlexChart** プロパティ名（axis-xまたはaxis-y）を定義します。
axis-line @ 軸線が表示されているかどうかを示す値。
binding @ 軸ラベルで使用する**itemsSource** プロパティの カンマ区切りのプロパティ名を取得または設定します。 最初の名前は軸の値を指定し、2番目は対応する軸ラベルを表します。デフォルト値は'value,text'です。
format @ 軸ラベルに使用される書式文字列 （**Globalize** を参照）。
item-formatter = 軸ラベルの外観をカスタマイズする書式設定関数。
items-source = 軸ラベルの項目ソース。
labels @ 軸ラベルが表示されているかどうかを示す値。
label-angle @ 軸ラベルの回転角度（度単位）。
label-align @ 軸ラベルの配置。
label-padding @ 軸ラベルのパディング。
major-grid @ 軸にグリッド線が含まれているかどうかを示す値。
major-tick-marks @ 軸の目盛りマークの外観を定義します （**TickMark** を参照）。
major-unit @ 軸ラベル間の単位数。
max @ 軸に表示される最小値。
min @ 軸に表示される最大値。
minor-grid @ 軸に副グリッド線が含まれているかどうかを示す値。
minor-tick-marks @ 軸の小目盛りマークの外観を定義します （**TickMark** を参照）。
minor-unit @ 小軸目盛り間の単位数。
origin @ 軸の原点。
overlappingLabels @ 重なった軸ラベルの処理方法を示す**OverlappingLabels** 値。
position @ 軸の位置を示す**Position** 値。
reversed @ 軸が反転（上から下または右から左）している かどうかを示す値。
title @ 軸の横に表示されるタイトルテキスト。

WjFlexChartBollingerBands クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart BollingerBands オブジェクトのAngularJSディレクティブ。

wj-flex-chart-bollinger-bandsディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
period @ ボリンジャーバンド計算の期間。
multiplier @ ボリンジャーバンド計算の標準偏差乗数。

WjFlexChartBoxWhisker クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart および**FinancialChart BoxWhisker** オブジェクト用のAngularJSディレクティブ。

wj-flex-chart-box-whiskerディレクティブは、**WjFlexChart** または**WjFinancialChart** ディレクティブに入れる必要があります。 次の属性をサポートします。

binding	@ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x	@ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
items-source	= この系列のデータを含む配列または**ICollectionView** オブジェクト。
name	@ 凡例に表示する系列の名前。
visibility	= 系列を表示するかどうかと、その場所を示す**SeriesVisibility** 値。
quartile-calculation	@ 箱ひげ図の四分位数計算を指定する値。
group-width	@ 箱ひげ図のグループ幅をパーセント値で指定する値。
gap-width	@ 箱ひげ図のギャップ幅をパーセント値で指定する値。
show-mean-line	@ 箱ひげ図の平均線を表示するかどうかを指定する値。
mean-line-style	@ 平均線のスタイルを指定する値。
show-mean-marker	@ 箱ひげ図の平均マーカーを表示するかどうかを指定する値。
mean-marker-style	@ 平均マーカーのスタイルを指定する値。
show-inner-points	@ 箱ひげ図の内側ポイントを表示するかどうかを指定する値。
show-outliers	@ 箱ひげ図の異常値を表示するかどうかを指定する値。

WjFlexChartCci クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart CCI オブジェクトのAngularJSディレクティブ。

wj-flex-chart-cciディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
period @ CCI（商品チャンネル指数）計算の期間。

WjFlexChartChartGestures クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart ChartGestures オブジェクトのAngularJSディレクティブ。

wj-flex-chart-gesturesディレクティブは、**WjFlexChart** ディレクティブまたは**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

mouse-action @ マウス操作がズームかパンかを示す値。
interactive-axes @ どの軸が操作可能かを示す値。
enable @ ジェスチャ操作が有効かどうかを示す値。
scale-x @ X軸の最小値から最大値までの初期範囲を示す値。
scale-y @ Y軸の最小値から最大値までの初期範囲を示す値。
pos-x @ X軸の初期位置を示す値。
pos-y @ Y軸の初期位置を示す値。

WjFlexChartDataLabel クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart の**DataLabel** オブジェクト用のAngularJSディレクティブ。

wj-flex-chart-data-labelディレクティブは、**WjFlexChart** ディレクティブ内に記述する必要があります。 このディレクティブは以下の属性をサポートしています。

content = データラベルを表す文字列、またはデータラベルの内容を取得または設定する関数。
border @ データラベルが境界線を持つかどうかを示す値を取得または設定します。
position @ データラベルの位置を示す**LabelPosition** 値。

WjFlexChartDataPoint クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart DataPoint オブジェクトのAngularJSディレクティブ。

wj-flex-chart-data-pointディレクティブは、**WjFlexChartAnnotation** ディレクティブに含める必要があります。**wj-flex-data-point**によって値が割り当てられる親ディレクティブのオブジェクトのプロパティは、**wj-property**属性で指定されます。

次の属性をサポートします。

wj-property @ **DataPoint**が割り当てられる親ディレクティブオブジェクトのプロパティの名前。
x @ x座標。数値または日付値を使用できます。
y @ y座標。数値または日付値を使用できます。

WjFlexChartEnvelopes クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart Envelopes オブジェクトのAngularJSディレクティブ。

wj-flex-chart-envelopesディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
period @ 移動平均エンベロープ計算の期間。
size @ 移動平均エンベロープのサイズ。
type @ エンベロープに使用する移動平均の **MovingAverageType**。

WjFlexChartErrorBar クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
基本クラス **WjFlexChartSeries**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart ErrorBar オブジェクト用のAngularJSディレクティブ。

wj-flex-chart-error-barディレクティブは、**WjFlexChart** ディレクティブに入れる必要があります。 次の属性をサポートします。

binding	@ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x	@ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
items-source	= この系列のデータを含む配列または**ICollectionView** オブジェクト。
name	@ 凡例に表示する系列の名前。
visibility	= 系列を表示するかどうかと、その場所を示す**SeriesVisibility** 値。
error-bar-style	@ 誤差範囲のスタイルを指定する値。
value	@ 系列の誤差値を指定する値。
error-amount	@ 系列の誤差量を指定する値。
end-style	@ 系列の終点スタイルを指定する値。
direction	@ 系列の方向を指定する値。

WjFlexChartFibonacci クラス

ファイル `wijmo.angular.js`
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart Fibonacci オブジェクトのAngularJSディレクティブ。

wj-flex-chart-fibonacciディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
high @ **Fibonacci** ツールの高値。
labelPosition @ **Fibonacci** ツールのレベルのラベル位置。
levels @ **Fibonacci** ツールのレベル値。
low @ **Fibonacci** ツールの低値。
minX @ **Fibonacci** ツールのx最小値。
maxX @ **Fibonacci** ツールのx最大値。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
altStyle = 系列の代替スタイル。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
uptrend @ 上昇トレンドの**Fibonacci** ツールを作成するかどうかを示す値。

WjFlexChartFibonacciArcs クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart FibonacciArcs オブジェクトのAngularJSディレクティブ。

wj-flex-chart-fibonacci-arcsディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
labelPosition @ **FibonacciArcs** ツールのレベルの**LabelPosition**。
levels @ **FibonacciArcs** ツールのレベル値。
start-x @ **FibonacciArcs** ツールの開始X値。
end-x @ **FibonacciArcs** ツールの終了X値。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
altStyle = 系列の代替スタイル。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。

WjFlexChartFibonacciFans クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart FibonacciFans オブジェクトのAngularJSディレクティブ。

wj-flex-chart-fibonacci-fansディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
labelPosition @ **FibonacciFans** ツールのレベルの**LabelPosition**。
levels @ **FibonacciFans** ツールのレベル値。
start @ **FibonacciFans** ツールの開始**DataPoint** 。
end @ **FibonacciFans** ツールの終了**DataPoint** 。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
altStyle = 系列の代替スタイル。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。

WjFlexChartFibonacciTimeZones クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart FibonacciTimeZones オブジェクトのAngularJSディレクティブ。

wj-flex-chart-fibonacci-time-zonesディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
labelPosition @ **FibonacciTimeZones** ツールのレベルの**LabelPosition**。
levels @ **FibonacciTimeZones** ツールのレベル値。
startX @ **FibonacciTimeZones** ツールの開始X値。
endX @ **FibonacciTimeZones** ツールの終了X値。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
altStyle = 系列の代替スタイル。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。

WjFlexChartLegend クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart の**Legend** オブジェクト用のAngularJSディレクティブ。

wj-flex-chart-legendディレクティブは、**WjFlexChart**、**WjFlexPie**、または**WjFinancialChart** ディレクティブ内に記述する必要があります。 このディレクティブは以下の属性をサポートしています。

position @ 凡例の位置を示す**Position** 値。

以下の例は、wj-flex-chart-legendディレクティブを使用してチャートの凡例の位置を変更する方法を示します。

```
<wj-flex-chart
  items-source="data"
  binding-x="country">
  <wj-flex-chart-axis
      wj-property="axisY"
      major-unit="5000">
    </wj-flex-chart-axis>
    <wj-flex-chart-series
      binding="sales"
      name="Sales">
    </wj-flex-chart-series>
  <wj-flex-chart-legend
    position="Bottom">
  </wj-flex-chart-legend>
</wj-flex-chart>
```

WjFlexChartLineMarker クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart LineMarker オブジェクトのAngularJSディレクティブ。

wj-flex-line-markerディレクティブは、**WjFlexChart** または**WjFinancialChart** ディレクティブに入れる必要があります。 このディレクティブは以下の属性をサポートしています。

is-visible	@ LineMarkerが表示されるかどうかを示す値。
series-index	@ LineMarkerが表示されるチャート内の系列のインデックス。
horizontal-position	@ プロットエリアに対するLineMarkerの水平位置。
content	@ LineMarkerのテキストコンテンツをカスタマイズするための関数。
vertical-position	@ プロットエリアに対するLineMarkerの垂直位置。
alignment	@ LineMarkerのコンテンツの配置を示す**LineMarkerAlignment** 値。
lines	@ LineMarkerの線の外観を示す**LineMarkerLines** 値。
interaction	@ LineMarkerの操作モードを示す**LineMarkerInteraction** 値。
drag-threshold	@ マーカーをドラッグできる水平線または垂直線からの最大距離。
drag-content	@ 操作モードがDragの場合に、マーカーのコンテンツをドラッグできるかどうかを示す値。
drag-lines	@ 操作モードがDragの場合に、水平線または垂直線がドラッグされたときに線がリンクされるかどうかを示す値。

WjFlexChartMacd クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart Macd オブジェクトのAngularJSディレクティブ。

wj-flex-chart-macdディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding	@ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x	@ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class	@ 系列に使用するCSSクラス。
items-source	= この系列のデータを含む配列または**ICollectionView** オブジェクト。
name	@ 凡例に表示する系列の名前。
style	= 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
styles	MACDラインとシグナルラインのスタイル。
symbol-marker	@ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size	@ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style	= Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility	= 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
fast-period	@ MACD計算の高速移動平均期間。
slow-period	@ MACD計算の低速移動平均期間。
signal-smoothing-period	@ MACD計算の平滑期間。

WjFlexChartMacdHistogram クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart MacdHistogram オブジェクトのAngularJSディレクティブ。

wj-flex-chart-macd-histogramディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding	@ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x	@ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class	@ 系列に使用するCSSクラス。
items-source	= この系列のデータを含む配列または**ICollectionView** オブジェクト。
name	@ 凡例に表示する系列の名前。
style	= 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker	@ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size	@ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style	= Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility	= 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
fast-period	@ MACD計算の高速移動平均期間。
slow-period	@ MACD計算の低速移動平均期間。
signal-smoothing-period	@ MACD計算の平滑期間。

WjFlexChartMovingAverage クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart および**FinancialChart MovingAverage** オブジェクトのAngularJSディレクティブ。

wj-flex-chart-moving-averageディレクティブは、**WjFlexChart** または**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
chart-type @ この系列オブジェクトに対応するオブジェクトのレンダリング時に使用するチャートタイプ。この値は、チャートに設定されたデフォルトのチャートタイプをオーバーライドします。**ChartType** を参照してください。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
type @ 移動平均系列の**MovingAverageType** 値。
period @ 移動平均計算の期間。

WjFlexChartParametricFunctionSeries クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart および**FinancialChart WjFlexChartParametricFunctionSeries** オブジェクト用のAngularJSディレクティブ。

wj-flex-chart-parametric-function-seriesディレクティブは、**WjFlexChart** ディレクティブまたは**WjFinancialChart** ディレクティブに含まれる必要があります。 次の属性をサポートします。

binding	@ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x	@ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
chart-type	@ この系列オブジェクトに対応するオブジェクトのレンダリング時に使用するチャートタイプ。この値は、チャートに設定されたデフォルトのチャートタイプをオーバーライドします。**ChartType** を参照してください。
css-class	@ 系列に使用するCSSクラス。
items-source	= この系列のデータを含む配列または**ICollectionView** オブジェクト。
name	@ 凡例に表示する系列の名前。
style	= 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker	@ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size	@ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style	= Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility	= 系列を表示するかどうか、表示する場合はその場所を示す**SeriesVisibility** 値。
sample-count	@ 計算のためのサンプル数。
min	@ 関数を計算するためのパラメータの最小値。
max	@ 関数を計算するためのパラメータの最大値。
x-func	@ x値の計算に使用される関数。
y-func	@ y値の計算に使用される関数。

WjFlexChartRangeSelector クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart RangeSelector オブジェクトのAngularJSディレクティブ。

wj-flex-chart-range-selectorディレクティブは、**WjFlexChart** ディレクティブまたは**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

is-visible @ RangeSelectorが表示されるかどうかを示す値。
min @ 範囲の最小値。
max @ 範囲の最大値。
orientation @ RangeSelectorの向き。
seamless @最小/最大ハンドラがシームレスに移行するかどうかを示す値。
min-scale @ RangeSelectorの有効な最小範囲。
max-scale @ RangeSelectorの有効な最大範囲。

WjFlexChartRsi クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart RSI オブジェクトのAngularJSディレクティブ。

wj-flex-chart-rsiディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
period @ RSI（相対力指数）計算の期間。

WjFlexChartSeries クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
派生クラス **WjFlexChartErrorBar**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart Series オブジェクトのAngularJSディレクティブ。

wj-flex-chart-seriesディレクティブは、**WjFlexChart** ディレクティブに含める必要があります。 このディレクティブは以下の属性をサポートしています。

axis-x @ 系列のX軸。
axis-y @ 系列のY軸。
binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
chart-type @ この系列オブジェクトに対応するオブジェクトのレンダリング時に使用するチャートタイプ。この値は、チャートに設定されたデフォルトのチャートタイプをオーバーライドします。**ChartType** を参照してください。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
altStyle = 系列の代替スタイル。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。

通常、**wj-flex-chart-series**では、**name**プロパティと**binding**プロパティだけを指定します。 残りの値は、親**wj-flex-chart**ディレクティブから継承されます。

WjFlexChartStochastic クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart Stochastic オブジェクトのAngularJSディレクティブ。

wj-flex-chart-stochasticディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding	@ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x	@ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class	@ 系列に使用するCSSクラス。
items-source	= この系列のデータを含む配列または**ICollectionView** オブジェクト。
name	@ 凡例に表示する系列の名前。
style	= 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
styles	%Kラインと%Dラインのスタイル。
symbol-marker	@ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size	@ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style	= Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility	= 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
k-period	@ %K計算の期間。
d-period	@ %D計算の期間。
smoothing-period	@ %K計算の平滑期間。

WjFlexChartTrendLine クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart および**FinancialChart TrendLine** オブジェクトのAngularJSディレクティブ。

wj-flex-chart-trend-lineディレクティブは、**WjFlexChart** または**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
chart-type @ この系列オブジェクトに対応するオブジェクトのレンダリング時に使用するチャートタイプ。この値は、チャートに設定されたデフォルトのチャートタイプをオーバーライドします。**ChartType** を参照してください。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、SplineSymbolsの各チャートでこの系列の データポイントのレンダリングに使用するシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
sample-count @ 計算のサンプル数。
fit-type @ 傾向線の**TrendLineFitType** 値。
order @ 多項式またはフーリエ方程式の項の数。

WjFlexChartWaterfall クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart および**FinancialChart Waterfall** オブジェクト用のAngularJSディレクティブ。

wj-flex-chart-waterfallディレクティブは、**WjFlexChart** ディレクティブまたは**WjFinancialChart** ディレクティブに含まれる必要があります。 次の属性をサポートします。

binding	@ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x	@ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
items-source	= この系列のデータを含む配列または**ICollectionView** オブジェクト。
name	@ 凡例に表示する系列の名前。
visibility	= 系列を表示するかどうか、表示する場合はその場所を示す**SeriesVisibility** 値。
relative-data	@ 指定されたデータが相対データかどうかを判定する値。
start	@ 開始バーの値。
start-label	@ 開始バーのラベル。
show-total	@ 合計バーを表示するかどうかを判定する値。
total-label	@ 合計バーのラベル。
show-intermediate-total	@ 小計バーを表示するかどうかを判定する値。
intermediate-total-positions	@ 小計バーの位置のインデックスを含む値。
intermediate-total-labels	@ 小計バーのラベルを含む値。
connector-lines	@ 接続線を表示するかどうかを判定する値。
スタイル	@ ウォータフォールスタイルの値。

WjFlexChartWilliamsR クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart WilliamsR オブジェクトのAngularJSディレクティブ。

wj-flex-chart-williams-rディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はどこに表示するかを示す**SeriesVisibility** 値。
period @ Williams %R計算の期間。

WjFlexChartYFunctionSeries クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart および**FinancialChart YFunctionSeries** オブジェクト用のAngularJSディレクティブ。

wj-flex-chart-y-function-seriesディレクティブは、**WjFlexChart** ディレクティブまたは**WjFinancialChart** ディレクティブに含まれる必要があります。 次の属性をサポートします。

binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
chart-type @ この系列オブジェクトに対応するオブジェクトのレンダリング時に使用するチャートタイプ。この値は、チャートに設定されたデフォルトのチャートタイプをオーバーライドします。**ChartType** を参照してください。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、およびSplineSymbolsチャートでこの系列のデータポイントのレンダリングに使用されるシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうか、表示する場合はその場所を示す**SeriesVisibility** 値。
sample-count @ 計算のためのサンプル数。
min @ 関数を計算するためのパラメータの最小値。
max @ 関数を計算するためのパラメータの最大値。
func @ Y値の計算に使用される関数。

WjFlexGrid クラス

ファイル	wijmo.angular.js
モジュール	**wijmo.angular**
派生クラス	**WjFlexSheet**, **WjMultiRow**, **WjPivotGrid**
表示	☐継承されたメンバー ☐イベント発生元

FlexGrid コントロール用のAngularJSディレクティブ。

wj-flex-gridディレクティブを使用して、AngularJSアプリケーションにグリッドを追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>FlexGridコントロールの例：</p>
<wj-flex-grid items-source="data">
  <wj-flex-grid-column
    header="Country"
    binding="country">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Sales"
    binding="sales">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Expenses"
    binding="expenses">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Downloads"
    binding="downloads">
  </wj-flex-grid-column>
</wj-flex-grid>
```

この例では、FlexGridコントロールを作成し、それをコントローラから公開されている'data'配列に連結しています。グリッドには3つの列があり、それぞれが ソース配列に含まれるオブジェクトの1つのプロパティに対応しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/QNb9X)

wj-flex-gridディレクティブは、次の属性をサポートします。

allow-add-new	@ ユーザーがソースコレクションに項目を追加できるように、 新しい行テンプレートを表示するかどうかを示す値。
allow-delete	@ ［Del］キーが押されたときに、グリッドで 選択されている行を削除するかどうかを示す値。
allow-dragging	@ ユーザーがマウスを使用して行と列をドラッグできるかどうかと、 その方法を示す**AllowDragging** 値。
allow-merging	@ グリッドのどの部分がセル結合を提供するかを 示す**AllowMerging** 値。
allow-resizing	@ ユーザーがマウスを使用して行と列を サイズ変更できるかどうかを示す**AllowResizing** 値。
allow-sorting	@ ユーザーが列ヘッダーをクリックして 列をソートできるかどうかを示すboolean値。
auto-generate-columns	@ グリッドが**items-source**に基づいて 列を自動的に生成するかどうかを示すboolean値。
child-items-path	@ 階層グリッドで子の行を生成するために使用される プロパティの名前（異なる階層レベルにある項目の子項目に異なる名前を 使用する場合は、プロパティ名の配列）。
control	= このディレクティブによって作成された **FlexGrid** コントロールへの参照。
defer-resizing	= ユーザーがマウスボタンを放すまで行と列のサイズ変更を 遅延するかどうかを示すboolean値。
frozen-columns	@ グリッド内の固定（スクロール不可能）列の数。
frozen-rows	@ グリッド内の固定（スクロール不可能）行の数。
group-header-format	@ グループヘッダーコンテンツの作成に使用される 書式文字列。
headers-visibility	= 行ヘッダーと列ヘッダーを表示するかどうかを示す**HeadersVisibility** 値。
ime-enabled	@編集モードでないときに、グリッドがIME（Input Method Editor）を サポートするかどうかを決定する値を取得または設定します。
initialized	& このイベントは、属性値でコントロールを初期化して連結処理が完了した後に発生します。
is-initialized	= 属性値でコントロールを初期化して連結処理が完了したかどうかを示す値。
item-formatter	= このグリッドのセルをカスタマイズする関数。
items-source	= グリッドに表示される項目を含む配列または**ICollectionView** オブジェクト。
is-read-only	@ ユーザーがグリッドセルにキー入力して編集でき ないようにされているかどうかを示すboolean値。
merge-manager	= 指定されたセルの結合範囲を指定する**MergeManager** オブジェクト。
selection-mode	@ ユーザーがセルを選択できるかどうかと、 その方法を示す**SelectionMode** 値。
show-groups	@ グループ行を挿入してデータグループを区切るかどうかを 示すboolean値。
show-sort	@ 列ヘッダーにソートインジケータを表示するかどうかを 示すboolean値。
sort-row-index	@ 現在のソートを表示および変更する、列ヘッダーパネル内の 行のインデックスを指定する数値。
tree-indent	@ 異なるレベルの行グループをオフセットするために 使用されるインデント（ピクセル単位）。
beginning-edit	& **beginningEdit** イベントのハンドラ。

cell-edit-ended	& **cellEditEnded** イベントのハンドラ。
cell-edit-ending	& **cellEditEnding** イベントのハンドラ。
prepare-cell-for-edit	& **prepareCellForEdit** イベントのハンドラ。
resizing-column	& **resizingColumn** イベントのハンドラ。
resized-column	& **resizedColumn** イベントのハンドラ。
dragged-column	& **draggedColumn** イベントのハンドラ。
dragging-column	& **draggingColumn** イベントのハンドラ。
sorted-column	& **sortedColumn** イベントのハンドラ。
sorting-column	& **sortingColumn** イベントのハンドラ。
deleting-row	& **deletingRow** イベントのハンドラ。
dragging-row	& **draggingRow** イベントのハンドラ。
dragged-row	& **draggedRow** イベントのハンドラ。
resizing-row	& **resizingRow** イベントのハンドラ。
resized-row	& **resizedRow** イベントのハンドラ。
row-added	& **rowAdded** イベントのハンドラ。
row-edit-ended	& **rowEditEnded** イベントのハンドラ。
row-edit-ending	& **rowEditEnding** イベントのハンドラ。
loaded-rows	& **loadedRows** イベントのハンドラ。
loading-rows	& **loadingRows** イベントのハンドラ。
group-collapsed-changed	& **groupCollapsedChanged** イベントのハンドラ。
group-collapsed-changing	& **groupCollapsedChanging** イベントのハンドラ。
items-source-changed	& **itemsSourceChanged** イベントのハンドラ。
selection-changing	& **selectionChanging** イベントのハンドラ。
selection-changed	& **selectionChanged** イベントのハンドラ。
got-focus	& **gotFocus** イベントのハンドラ。
lost-focus	& **lostFocus** イベントのハンドラ。
scroll-position-changed	& **scrollPositionChanged** イベントのハンドラ。

wj-flex-gridディレクティブには、**WjFlexGridColumn**、**WjFlexGridCellTemplate**、 **WjFlexGridDetail** の各子ディレクティブを入れることができます。

WjFlexGridCellTemplate クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid セルテンプレート用のAngularJSディレクティブ。

wj-flex-grid-cell-templateは**FlexGrid** の特定のセルタイプ用のテンプレートを定義するディレクティブであり、**CellTemplateType** を指定する**cell-type**属性を含める必要があります。テンプレートのセルタイプに応じて、**wj-flex-grid-cell-template**ディレクティブは**WjFlexGrid** または**WjFlexGridColumn** ディレクティブの子にする必要があります。

列固有のセルテンプレートは**wj-flex-grid-column**ディレクティブに含める必要があり、列固有でないセル（行ヘッダセルや左上セルなど）のテンプレートは**wj-flex-grid**ディレクティブに含める必要があります。

wj-flex-grid-cell-templateディレクティブには、HTMLフラグメントに加えて、セルの条件付き書式を提供する**ng-style**または**ng-class**属性を含めることができます。

ng-style/ng-class属性とHTMLフラグメントのどちらにおいても、**$col**、**$row**、**$item**の各テンプレート変数を使用できます。これらはそれぞれ、そのセルに関係する**Column** オブジェクト、**Row** オブジェクト、**Row.dataItem**オブジェクトを参照します。

Groupや**CellEdit**などのセルタイプでは、書式設定されていないセル値を含む**$value**変数も提供されます。たとえば、以下のFlexGridコントロールには行ヘッダ用のテンプレートとCountry列の通常セルおよび列ヘッダセル用のテンプレートが設定されています。

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="RowHeader">
    {{$row.index}}
  </wj-flex-grid-cell-template>
  <wj-flex-grid-cell-template cell-type="RowHeaderEdit">
    ...
  </wj-flex-grid-cell-template>

  <wj-flex-grid-column header="Country" binding="country">
    <wj-flex-grid-cell-template cell-type="ColumnHeader">
      <img ng-src="resources/globe.png" />
        {{$col.header}}
      </wj-flex-grid-cell-template>
      <wj-flex-grid-cell-template cell-type="Cell">
        <img ng-src="resources/{{$item.country}}.png" />
        {{$item.country}}
      </wj-flex-grid-cell-template>
    </wj-flex-grid-column>
  <wj-flex-grid-column header="Sales" binding="sales"></wj-flex-grid-column>
</wj-flex-grid>
```

特定のセルタイプ用のテンプレートの詳細については、**CellTemplateType** 列挙体のドキュメントを参照してください。

wj-flex-grid-columnディレクティブには、通常のデータセル（*cell-type="Cell"*）用のテンプレートして扱われる任意のコンテンツを含めることもできます。ただし、**wj-flex-grid-column**ディレクティブ内に、*cell-type="Cell"*に設定された**wj-flex-grid-cell-template**ディレクティブが存在する場合は、このテンプレートの方が優先され、任意のコンテンツはオーバーライドされます。

wj-flex-grid-cell-templateディレクティブは以下の属性をサポートしています。

cell-type @ テンプレートを適用するセルのタイプを定義する**CellTemplateType** 値。
cell-overflow @ セルの**style.overflow**プロパティの値を定義します。

force-full-edit
@ セル編集テンプレートでは、編集がどのように開始されたかにかかわらず、 セル編集が完全編集モードで強制的に開始するかどうかを示します。 完全編集モードでは、カーソルキーを押しても編集が終了しません。 デフォルトはtrueです。

cell-type属性には以下のいずれかの列挙値を指定します。

Cell 通常の（データ）セル用のテンプレートを定義します。**WjFlexGridColumn** ディレクティブの子にする必要があります。 たとえば、以下のセルテンプレートはCountry列のセルに国旗を表示します。

```
<wj-flex-grid-column header="Country" binding="country">
  <wj-flex-grid-cell-template cell-type="Cell">
    <img ng-src="resources/{{$item.country}}.png" />
    {{$item.country}}
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

階層的な**FlexGrid** （すなわち、**childItemsPath**プロパティが指定されているグリッド）では、**Group**テンプレートが提供されていない場合、この**Column**のグループ行に含まれるヘッダ以外のセルにもこのテンプレートが適用されます。

CellEdit

編集モードのセル用のテンプレートを定義します。**WjFlexGridColumn** ディレクティブの子にする必要があります。 このセルタイプでは、追加の**$value**プロパティを使用できます。このプロパティには編集前のセル値が格納され、編集後は値が更新されます。 たとえば、以下のテンプレートは"Sales"列のエディタとしてWijmo **InputNumber** コントロールを使用します。

```
<wj-flex-grid-column header="Sales" binding="sales">
  <wj-flex-grid-cell-template cell-type="CellEdit">
    <wj-input-number value="$value" step="1"></wj-input-number>
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

ColumnHeader

列ヘッダセル用のテンプレートを定義します。**WjFlexGridColumn** ディレクティブの子にする必要があります。 たとえば、以下のテンプレートは"Country"列のヘッダに画像を追加します。

```
<wj-flex-grid-column header="Country" binding="country">
  <wj-flex-grid-cell-template cell-type="ColumnHeader">
    <img ng-src="resources/globe.png" />
    {{$col.header}}
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

RowHeader

行ヘッダセル用のテンプレートを定義します。**WjFlexGrid** ディレクティブの子にする必要があります。 たとえば、以下のテンプレートは行ヘッダに行インデックスを表示します。

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="RowHeader">
    {{$row.index}}
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

このテンプレートは、編集モードにある行の行ヘッダセルにも適用されます。編集モードの行ヘッダセルに別のコンテンツを提供するには、**RowHeaderEdit**テンプレートを定義します。

RowHeaderEdit

編集モードの行ヘッダセル用のテンプレートを定義します。**WjFlexGrid** ディレクティブの子にする必要があります。たとえば、以下のテンプレートは編集中の行のヘッダに点を表示します。

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="RowHeaderEdit">
      ...
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

RowHeaderテンプレートを適用するセルに標準の編集モードインジケーターを追加するには、以下の**RowHeaderEdit**テンプレートを使用します。

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="RowHeaderEdit">
    {{&#x270e;}}
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

TopLeft

左上セル用のテンプレートを定義します。**WjFlexGrid** ディレクティブの子にする必要があります。 たとえば、以下のテンプレートはグリッドの左上セルに右下向きの矢印を表示します。

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="TopLeft">
    <span class="wj-glyph-down-right"></span>
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

GroupHeader

GroupRow のグループヘッダセル用のテンプレートを定義します。 **WjFlexGridColumn** ディレクティブの子にする必要があります。

$row変数には、**GroupRow**クラスのインスタンスが格納されます。グループ化が **CollectionView** に由来する場合、**$item**変数は**CollectionViewGroup** オブジェクトを参照します。

たとえば、以下のテンプレートは展開/折りたたみ状態を切り替えるためにチェックボックス要素を使用します。

```
<wj-flex-grid-column header="Country" binding="country">
  <wj-flex-grid-cell-template cell-type="GroupHeader">
    <input type="checkbox" ng-model="$row.isCollapsed"/>
    {{$item.name}} ({{$item.items.length}} items)
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

Group

GroupRow の通常のセル（グループヘッダ以外のセル）用のテンプレートを定義します。 **WjFlexGridColumn** ディレクティブの子にする必要があります。このセルタイプでは、追加の**$value**変数を使用できます。列に**aggregate**プロパティが指定されている場合、$value変数には書式設定されていない集計値が格納されます。

たとえば、以下のテンプレートは"Sales"列のグループ行のセルに集計値と集計の種類を表示します。

```
<wj-flex-grid-column header="Sales" binding="sales" aggregate="Avg">
  <wj-flex-grid-cell-template cell-type="Group">
    Average:{{$value | number:2}}
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

ColumnFooter

columnFootersパネル内の通常のセルのテンプレートを定義します。**WjFlexGridColumn** ディレクティブの子にする必要があります。このセルタイプでは、セル値を含むバインディン用に追加の**$value**プロパティを使用できます。

たとえば、以下のテンプレートは"Sales"列のフッターのセルに集計値と集計の種類を表示します。

```
<wj-flex-grid-column header="Sales" binding="sales" aggregate="Avg">
  <wj-flex-grid-cell-template cell-type="ColumnFooter">
    Average:{{$value | number:2}}
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

BottomLeft

左下のセルのテンプレートを定義します（行ヘッダーと列フッターが交差する位置にあるセル）。 **WjFlexGrid** ディレクティブの子にする必要があります。たとえば、このテンプレートは、グリッドの左下のセルにシグマの矢印を表示します。

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="BottomLeft">
   &#931;
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

WjFlexGridColumn クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

Column オブジェクト用のAngularJSディレクティブ。

wj-flex-grid-columnディレクティブは、**WjFlexGrid** ディレクティブ内に記述する必要があります。 次の属性をサポートします。

aggregate	@ この列のグループヘッダ行に表示する **Aggregate** オブジェクト。
align	@ 列の項目の水平方向の配置を左、右、または中央に設定する文字列値。
allow-dragging	@ ユーザーがマウスで列を新しい位置に移動できるかどうかを示す値。
allow-sorting	@ ユーザーが列ヘッダをクリックして列をソートできるかどうかを示す値。
allow-resizing	@ ユーザーがマウスで列のサイズを変更できるかどうかを示す値。
allow-merging	@ ユーザーが列のセルを結合できるかどうかを示す値。
binding	@ 列がバインドされているプロパティの名前。
css-class	@ 列をレンダリングするときに使用するCSSクラスの名前。
data-map	= 生の値からこの列の表示値への変換に使用する**DataMap** オブジェクト。
data-type	@ 列に格納されるデータの型を示す**DataType** 列挙値。
format	@ 生の値からこの列の表示値への変換に使用する書式文字列（**Globalize** を参照）。
header	@ 列ヘッダに表示する文字列。
input-type	@ 列の値の編集に使用される入力要素を指定するtype属性。デフォルトは、数値型の列では"tel"、その他のブール型以外の列では"text"です。
is-content-html	@ この列のセルの内容がプレーンテキストではなくHTMLコンテンツであるかどうかを示す値。
is-read-only	@ ユーザーが列の値を編集できないようにするかどうかを示す値。
is-selected	@ 列が選択されているかどうかを示す値。
mask	@ 列の値の編集に使用されるマスク文字列。
max-width	@ 列の最大幅。
min-width	@ 列の最小幅。
name	@ 列名。これを使用して列を取得できます。
is-required	@ 列の値を必ずnull以外にしなければならないかどうかを示す値。
show-drop-down	@ 列の**DataMap** に基づいて編集するためのドロップダウンボタンを表示するかどうかを示す値。
visible	@ 列が表示されるかどうかを示す値。
width	@ 列の幅（ピクセル単位またはスター値）。
word-wrap	@ 列のセルの内容を折り返すかどうかを示す値。

wj-flex-grid-columnディレクティブ内のHTMLコンテンツは、その列のセル用のテンプレートして扱われます。 テンプレートは通常のセルのみに適用されます。列ヘッダやグループヘッダなどの特定のセルタイプにテンプレートを適用する場合は、 **WjFlexGridCellTemplate** ディレクティブを参照してください。

次の例では、テンプレートと条件付きスタイルを使用して2つの列を作成します。

サンプル

Show me (https://jsfiddle.net/Wijmo5/5L423)

wj-flex-grid-columnディレクティブには、子ディレクティブとして**WjFlexGridCellTemplate** を含めることができます。

WjFlexGridDetail クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid の**DetailRow** テンプレート用のAngularJSディレクティブ。

wj-flex-grid-detailディレクティブは、**wj-flex-grid**ディレクティブ内に記述する必要があります。

wj-flex-grid-detailディレクティブは、詳細行の表示/非表示設定を保持する**FlexGridDetailProvider** オブジェクトと、ディレクティブタグ内で任意のHTMLフラグメントとして定義された詳細行の内容を表します。このフラグメントにはAngularJSバインディングおよびディレクティブを含めることができます。コントローラーで使用可能なプロパティに加えて、ローカルの**$row**および**$item**テンプレート変数をAngularJSバインディングで使用して詳細行の親**Row**と**Row.dataItem**オブジェクトを参照できます。次に例を示します。

```
<p>FlexGridが入れ子になった詳細行:</p>

<wj-flex-grid
  items-source="categories">
  <wj-flex-grid-column header="Name" binding="CategoryName"></wj-flex-grid-column>
  <wj-flex-grid-column header="Description" binding="Description" width="*"></wj-flex-grid-column>
  <wj-flex-grid-detail max-height="250" detail-visibility-mode="detailMode">
    <wj-flex-grid
      items-source="getProducts($item.CategoryID)"
      headers-visibility="Column">
    </wj-flex-grid>
  </wj-flex-grid-detail>
</wj-flex-grid>
```

wj-flex-grid-detailディレクティブによって表される**FlexGridDetailProvider**オブジェクトへの参照は、通常どおりディレクティブの**control**プロパティにバインドすることによって取得できます。これにより、**FlexGridDetailProvider**によって提供されるすべてのAPIをテンプレートで使用できるため、ユーザーエクスペリエンスを完全にコントロールできます。以下の例では、Name列のセルにカスタムの表示/非表示トグルを追加し、詳細行に［Hide Detail］ボタンを追加します。これらの要素がクリックされたときに**ng-click**バインディングで指定した**FlexGridDetailProvider**の**hideDetail**および**showDetail**メソッドを呼び出すことで、カスタムの表示/非表示ロジックを実装しています。

```
<p>詳細行を表示/非表示にするカスタム要素を持つFlexGrid:</p>

<wj-flex-grid
  items-source="categories"
  headers-visibility="Column"
  selection-mode="Row">
  <wj-flex-grid-column header="Name" binding="CategoryName" is-read-only="true" width="200">
    <img ng-show="dp.isDetailVisible($row)" ng-click="dp.hideDetail($row)" src="resources/hide.png" />
    <img ng-hide="dp.isDetailVisible($row)" ng-click="dp.showDetail($row, true)" src="resources/show.png" />
    {{$item.CategoryName}}
  </wj-flex-grid-column>
  <wj-flex-grid-column header="Description" binding="Description" width="2*"></wj-flex-grid-column>
  <wj-flex-grid-detail control="dp" detail-visibility-mode="Code">
    <div style="padding:12px;background-color:#cee6f7">
      ID:<b>{{$item.CategoryID}}</b><br />
      Name:<b>{{$item.CategoryName}}</b><br />
      Description:<b>{{$item.Description}}</b><br />
      <button class="btn btn-default" ng-click="dp.hideDetail($row)">Hide Detail</button>
    </div>
  </wj-flex-grid-detail>
</wj-flex-grid>
```

wj-flex-grid-detailディレクティブは以下の属性をサポートしています。

wj-flex-grid-detailディレクティブは以下の属性をサポートしています。

control = このディレクティブによって作成された**FlexGridDetailProvider** オブジェクトへの参照。
detail-visibility-mode @ 行の詳細をいつ表示するかを決定する**DetailVisibilityMode** 値。
max-height @ 詳細行の最大の高さ（ピクセル単位）。
row-has-detail = 行が詳細を持つかどうかを決定するコールバック関数。

WjFlexGridFilter クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGridFilter オブジェクトのAngularJSディレクティブ。

wj-flex-grid-filterディレクティブは、**WjFlexGrid** ディレクティブに入れる必要があります。次に例を示します。

```
<p>列フィルタを含むFlexGridコントロールの例</p>
<wj-flex-grid items-source="data">
  <wj-flex-grid-filter filter-columns="['country', 'expenses']"></wj-flex-grid-filter>

  <wj-flex-grid-column
    header="Country"
    binding="country">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Sales"
    binding="sales">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Expenses"
    binding="expenses">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Downloads"
    binding="downloads">
  </wj-flex-grid-column>
</wj-flex-grid>
```

wj-flex-grid-filterディレクティブは、次の属性をサポートします。

filter-columns = フィルタ処理する列の名前または連結を含む配列。
show-filter-icons @ フィルタ編集ボタンをグリッドの列ヘッダーに 表示するかどうかを示す値。
filter-changing & **filterChanging** イベントのハンドラ。
filter-changed & **filterChanged** イベントのハンドラ。
filter-applied & **filterApplied** イベントのハンドラ。

WjFlexPie クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
派生クラス **WjSunburst**
表示 ☐継承されたメンバー ☐イベント発生元

FlexPie コントロール用のAngularJSディレクティブ。

items-source	=	チャートのデータを含む配列または**ICollectionView** オブジェクト。
binding	@	項目の値を含むプロパティの名前。
binding-name	@	項目の名前を含むプロパティの名前。
footer	@	チャートのフッタに表示するテキスト（プレーンテキスト）。
footer-style	=	チャートのフッタに適用するスタイル。
header	@	チャートのヘッダに表示するテキスト（プレーンテキスト）。
header-style	=	チャートのヘッダに適用するスタイル。
initialized	&	このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	=	バインディングによるコントロールの初期化が完了したかどうかを示す値。
inner-radius	@	パイの内側にある穴のサイズ（パイ半径に対する割合）。
is-animated	@	選択された項目をselectedItemPositionに移動するアニメーションを使用するかどうかを示す値。
item-formatter	=	データポイントの外観をカスタマイズするフォーマッター関数。
offset	@	パイスライスを中心から引き出す程度（パイ半径に対する割合）。
palette	=	パイスライスの表示に使用されるデフォルト色を含む配列。
plot-margin	=	コントロールの端からプロット領域の端までの間隔のピクセル数、またはCSSスタイルのマージン。
reversed	@	パイスライスを反時計回りに描画するかどうかを示す値。
start-angle	@	パイスライスの開始角度（9時の位置から時計回りに測定）。
selected-item-offset	@	選択されたパイスライスを中心から引き出す程度（パイ半径に対する割合）。
selected-item-position	@	選択されたスライスを表示する場所を示す**Position** 値。
selection-mode	@	ユーザーが系列をクリックしたときに何が選択されるかを示す**SelectionMode** 値。
tooltip-content	@	**ChartTooltip** に表示する内容。
got-focus	&	**gotFocus** イベントハンドラ。
lost-focus	&	**lostFocus** イベントハンドラ。
rendering	&	**rendering** イベントハンドラ。
rendered	&	**rendered** イベントハンドラ。

wj-flex-pieディレクティブには、子ディレクティブとして **WjFlexChartLegend** および**WjFlexPieDataLabel** を含めることができます。

WjFlexPieDataLabel クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexPie の**PieDataLabel** オブジェクト用のAngularJSディレクティブ。

wj-flex-pie-data-labelディレクティブは、**WjFlexPie** ディレクティブ内に記述する必要があります。 このディレクティブは以下の属性をサポートしています。

content = データラベルを表す文字列、またはデータラベルの内容を取得または設定する関数。
border @ データラベルが境界線を持つかどうかを示す値を取得または設定します。
position @ データラベルの位置を示す**PieLabelPosition** 値。

WjFlexRadar クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexRadar コントロール用のAngularJSディレクティブ。

wj-flex-radarディレクティブを使用して、AngularJSアプリケーションにレーダーチャートを追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。

wj-flex-radarディレクティブは、次の属性をサポートします。

binding	@ チャートのYの値を含むプロパティの名前。これは系列レベルでオーバーライドできます。
binding-x	@ チャートのXの値を含むプロパティの名前。これは系列レベルでオーバーライドできます。
chart-type	@ 系列オブジェクトのレンダリング時に使用するデフォルトのチャートタイプ。これは、系列レベルでオーバーライドできます。**RadarChartType** を参照してください。
control	= このディレクティブによって作成された **FlexRadar** コントロールへの参照。
footer	@ チャートフッターに表示するテキスト （プレーンテキスト）。
footer-style	= チャートフッターに適用するスタイル。
header	@ チャートヘッダーに表示するテキスト （プレーンテキスト）。
header-style	= チャートヘッダーに適用するスタイル。
initialized	& このイベントは、属性値でコントロールを初期化して 連結処理が完了した後に発生します。
is-initialized	= 属性値でコントロールを初期化して 連結処理が完了したかどうかを示す値。
interpolate-nulls	@ データにnull値がある場合に、 ギャップを補間するか残すかを示す値。
item-formatter	= データポイントの外観をカスタマイズする 書式設定関数。
items-source	= チャートの作成に使用されるデータを含む配列 または**ICollectionView** オブジェクト。
legend-toggle	@ 凡例項目をクリックしたときに系列の表示/非表示を 切り替えるかどうかを示す値。
options	= 特定のチャートタイプにのみ適用されるチャートの**options**。
palette	= 各系列の表示に使用されるデフォルトの色を 含む配列。
plot-margin	= コントロールの端からプロット領域までの ピクセル単位のスペース（CSS形式のマージン）。
stacking	@ 系列オブジェクトが積み重ねられているか、 個別にプロットされているかを示す**Stacking** 値。
reversed	@ 角度を反転（反時計回り）するかどうかを示す**reversed** 値。
startAngle	@ レーダーの開始角度（度単位）を示す**startAngle** 値。
totalAngle	@ レーダーの合計角度（度単位）を示す**totalAngle** 値。
symbol-size	@ Scatter、LineSymbols、SplineSymbolsの各チャートで データポイントのレンダリングに使用するシンボルのサイズ（ピクセル単位）。これは、系列レベルで オーバーライドできます。
tooltip-content	@ **ChartTooltip** コンテンツプロパティに 表示する値。
rendering	& **rendering** イベントのハンドラ。
rendered	& **rendered** イベントのハンドラ。
series-visibility-changed	& **seriesVisibilityChanged** イベントのハンドラ。

wj-flex-radarディレクティブは、 **WjFlexChartAxis**、**WjFlexRadarSeries**、**WjFlexChartLegend**、および**WjFlexChartDataLabel** を子ディレクティブとして含むことができます。

WjFlexRadarAxis クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FlexRadar FlexRadarAxis Axis オブジェクト用のAngularJSディレクティブ。

wj-flex-radar-axisディレクティブは、**WjFlexRadar** ディレクティブに含める必要があります。 次の属性をサポートします。

wj-property	@ このディレクティブで初期化する**FlexChart** プロパティ名（axis-xまたはaxis-y）を定義します。
axis-line	@ 軸線が表示されているかどうかを示す値。
binding	@ 軸ラベルで使用する**itemsSource** プロパティの カンマ区切りのプロパティ名を取得または設定します。 最初の名前は軸の値を指定し、2番目は対応する軸ラベルを表します。デフォルト値は'value,text'です。
format	@ 軸ラベルに使用される書式文字列 （**Globalize** を参照）。
item-formatter	= 軸ラベルの外観をカスタマイズする書式設定関数。
items-source	= 軸ラベルの項目ソース。
labels	@ 軸ラベルが表示されているかどうかを示す値。
label-angle	@ 軸ラベルの回転角度（度単位）。
label-align	@ 軸ラベルの配置。
label-padding	@ 軸ラベルのパディング。
major-grid	@ 軸にグリッド線が含まれているかどうかを示す値。
major-tick-marks	@ 軸の目盛りマークの外観を定義します （**TickMark** を参照）。
major-unit	@ 軸ラベル間の単位数。
max	@ 軸に表示される最小値。
min	@ 軸に表示される最大値。
minor-grid	@ 軸に副グリッド線が含まれているかどうかを示す値。
minor-tick-marks	@ 軸の小目盛りマークの外観を定義します （**TickMark** を参照）。
minor-unit	@ 小軸目盛り間の単位数。
origin	@ 軸の原点。
overlappingLabels	@ 重なった軸ラベルの処理方法を示す**OverlappingLabels** 値。
position	@ 軸の位置を示す**Position** 値。
title	@ 軸の横に表示されるタイトルテキスト。

WjFlexRadarSeries クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart FinancialSeries オブジェクトのAngularJSディレクティブ。

wj-financial-chart-seriesディレクティブは、**WjFinancialChart** ディレクティブに含める必要があります。 次の属性をサポートします。

axis-x @ 系列のX軸。
axis-y @ 系列のY軸。
binding @ 系列のY値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
binding-x @ 系列のX値を含むプロパティの名前。この値は、チャートに設定された連結をオーバーライドします。
chart-type @ この系列オブジェクトに対応するオブジェクトのレンダリング時に使用するチャートタイプ。この値は、チャートに設定されたデフォルトのチャートタイプをオーバーライドします。**FinancialChartType** を参照してください。
css-class @ 系列に使用するCSSクラス。
items-source = この系列のデータを含む配列または**ICollectionView** オブジェクト。
name @ 凡例に表示する系列の名前。
style = 系列スタイル。ng-attr-styleを使用して、系列スタイルオブジェクトをオブジェクトとして指定します。詳細については、AngularJSドキュメントの「**Creating Custom Directives**」にあるngAttr属性連結に関するセクションおよび「FlexChart 101：系列のスタイルの設定 (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling)」のサンプルを参照してください。
altStyle = 系列の代替スタイル。
symbol-marker @ 系列に使用するマーカーの形。この値は、チャートに設定されたデフォルトのマーカーをオーバーライドします。**Marker** を参照してください。
symbol-size @ Scatter、LineSymbols、SplineSymbolsの各チャートでこの系列の データポイントのレンダリングに使用するシンボルのサイズ（ピクセル単位）。 この値は、チャートレベルの設定をオーバーライドします。
symbol-style = Scatter、LineSymbols、SplineSymbolsの各チャートでこの系列の データポイントのレンダリングに使用するシンボルのスタイル。 この値は、チャートレベルの設定をオーバーライドします。
visibility = 系列を表示するかどうかと、その場所を示す**SeriesVisibility** 値。

通常、**wj-financial-chart-series**では、**name**プロパティと**binding**プロパティだけを指定します。 残りの値は、親**wj-financial-chart**ディレクティブから継承されます。

WjFlexSheet クラス

ファイル	wijmo.angular.js
モジュール	**wijmo.angular**
基本クラス	**WjFlexGrid**
表示	☐継承されたメンバー ☐イベント発生元

FlexSheet コントロール用のAngularJSディレクティブ。

wj-flex-sheetディレクティブは、AngularJSアプリケーションに**FlexSheet**コントロールを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>バインドされた1つのシートとバインドされていない2つのシートを含むFlexSheetコントロール:</p>
<wj-flex-sheet>
   <wj-sheet name="Country" items-source="ctx.data"></wj-sheet>
   <wj-sheet name="Report" row-count="25" column-count="13"></wj-sheet>
   <wj-sheet name="Formulas" row-count="310" column-count="10"></wj-sheet>
</wj-flex-sheet>
```

wj-flex-sheetディレクティブは**WjFlexGrid** を拡張したもので、以下の属性を持ちます。

control	= このディレクティブによって作成された**FlexSheet** コントロールへの参照。
is-tab-holder-visible	@ シートタブホルダーが表示されるかどうかを示す値。
selected-sheet-index	= **FlexSheet** で現在選択されているシートのインデックスを取得または設定します。
dragging-row-column	& **draggingRowColumn** イベントハンドラ。
dropping-row-column	& **droppingRowColumn** イベントハンドラ。
selected-sheet-changed	& **selectedSheetChanged** イベントハンドラ。

wj-flex-sheetディレクティブには、子ディレクティブとして**WjSheet** を含めることができます。

WjGroupPanel クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

GroupPanel コントロール用のAngularJSディレクティブ。

wj-group-panelディレクティブは、**grid**プロパティによって**FlexGrid**コントロールに接続します。 次に例を示します。

```
<p>GroupPanelを含むFlexGridコントロール:</p>

<wj-group-panel grid="flex" placeholder="Drag columns here to create groups."></wj-group-panel>

<wj-flex-grid control="flex" items-source="data">
  <wj-flex-grid-column
    header="Country"
    binding="country">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Sales"
    binding="sales">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Expenses"
    binding="expenses">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Downloads"
    binding="downloads">
  </wj-flex-grid-column>
</wj-flex-grid>
```

wj-group-panelディレクティブは以下の属性をサポートしています。

grid @ この**GroupPanel**に接続している**FlexGrid**。
hide-grouped-columns @ グループ化列をオーナーグリッドで非表示にするかどうかを示す値。
max-groups @ 許可されるグループの最大数。
placeholder @ グループがないときにコントロールに表示する文字列。
got-focus & **gotFocus** イベントハンドラ。
lost-focus & **lostFocus** イベントハンドラ。

WjInputColor クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

InputColor コントロール用のAngularJSディレクティブ。

wj-input-colorディレクティブは、AngularJSアプリケーションに**InputColor** コントロールを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>InputColorコントロール:</p>
<wj-input-color
  value="theColor"
  show-alpha-channel="false">
</wj-input-color>
```

wj-input-colorディレクティブは以下の属性をサポートしています。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによってバインドされるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用してスコープにバインドされるコントロールプロパティを指定します。
control	= このディレクティブによって作成されたInputColorコントロールへの参照。
is-dropped-down	@ ドロップダウンが現在表示されているかどうかを示す値。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
show-alpha-channel	@ ドロップダウンにアルファチャネル（透明度）エディタを表示するかどうかを示す値。
placeholder	@ コントロールが空のときにヒントとして表示する文字列。
is-required	@ null値が禁止されるかどうかを示す値。
show-drop-down-button	@ コントロールにドロップダウンボタンを表示するかどうかを示す値。
text	= コントロールに表示するテキスト。
value	= 編集する色。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。
is-dropped-down-changing	& **isDroppedDownChanging** イベントハンドラ。
is-dropped-down-changed	& **isDroppedDownChanged** イベントハンドラ。
text-changed	& **textChanged** イベントハンドラ。
value-changed	& **valueChanged** イベントハンドラ。

WjInputDate クラス

ファイル	wijmo.angular.js
モジュール	**wijmo.angular**
派生クラス	**WjInputDateTime**
表示	☐継承されたメンバー ☐イベント発生元

InputDate コントロール用のAngularJSディレクティブ。

wj-input-dateディレクティブを使用して、AngularJSアプリケーションに**InputDate** コントロールを追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>Here is an InputDate control:</p>
<wj-input-date
  value="theDate"
  format="M/d/yyyy">
</wj-input-date>
```

次の例は、**InputDate** コントロールと**InputTime** コントロールを使用して、（日時の情報を含む）**Date**値を表示します。どちらのコントロールも同じコントローラ変数に連結されており、各コントロールはそれぞれの情報（日付または時刻）を編集します。この例では、１回のクリックで日付を選択できる**Calendar** コントロールも示しています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/46PhD)

wj-input-dateディレクティブは、次の属性をサポートします。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによって連結されるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用して、スコープに連結するコントロールプロパティを指定します。
control	= このディレクティブによって作成された**InputDate** コントロールへの参照。
format	@ 編集中の日付の表示に使用される書式（**Globalize** を参照）。
mask	@ ユーザー入力時に入力の検証に使用されるマスク（**InputMask** を参照）。
is-dropped-down	@ ドロップダウンが現在表示されているかどうかを示す値。
initialized	& このイベントは、属性値でコントロールを初期化して連結処理が完了した後に発生します。
is-initialized	= 属性値でコントロールを初期化して連結処理が完了したかどうかを示す値。
max	@ 最も遅い有効な日付（書式"yyyy-MM-dd"の文字列）。
min	@ 最も早い有効な日付（書式"yyyy-MM-dd"の文字列）。
placeholder	@ コントロールが空のときにヒントとして表示する文字列。
is-required	@ null値を禁止するかどうかを示す値。
show-drop-down-button	@ コントロールにドロップダウンボタンを表示するかどうかを示す値。
text	= コントロールに表示するテキスト。
value	= 編集中の日付。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。
is-dropped-down-changing	& **isDroppedDownChanging** イベントハンドラ。
is-dropped-down-changed	& **isDroppedDownChanged** イベントハンドラ。
text-changed	& **textChanged** イベントハンドラ。
value-changed	& **valueChanged** イベントハンドラ。

min属性と**max**属性を指定する場合、それらは書式"yyyy-MM-dd"の文字列にする必要があります。技術的には、W3Cの**[RFC 3339]**で定義されている任意の日時を使用できます。これは、標準HTML5入力要素でも使用される書式です。

WjInputDateTime クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
基本クラス **WjInputDate**
表示 ☐継承されたメンバー ☐イベント発生元

InputDateTime コントロール用のAngularJSディレクティブ。

wj-input-date-timeディレクティブを使用して、AngularJSアプリケーションに**InputDateTime** コントロールを追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>Here is an InputDateTime control:</p>
<wj-input-date-time
  value="theDate"
  format="M/d/yyyy">
</wj-input-date-time>
```

wj-input-date-timeディレクティブは、次の属性をサポートします。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによって連結されるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用して、スコープに連結するコントロールプロパティを指定します。
control	= このディレクティブによって作成された**InputDate** コントロールへの参照。
format	@ 編集中の日付の表示に使用される書式（**Globalize** を参照）。
mask	@ ユーザー入力時に入力の検証に使用されるマスク（**InputMask** を参照）。
is-dropped-down	@ ドロップダウンが現在表示されているかどうかを示す値。
initialized	& このイベントは、属性値でコントロールを初期化して連結処理が完了した後に発生します。
is-initialized	= 属性値でコントロールを初期化して連結処理が完了したかどうかを示す値。
max	@ 最も遅い有効な日付（書式"yyyy-MM-dd"の文字列）。
min	@ 最も早い有効な日付（書式"yyyy-MM-dd"の文字列）。
placeholder	@ コントロールが空のときにヒントとして表示する文字列。
is-required	@ null値を禁止するかどうかを示す値。
show-drop-down-button	@ コントロールにドロップダウンボタンを表示するかどうかを示す値。
text	= コントロールに表示するテキスト。
timeMax	@ 最も早い有効な時刻（書式"hh:mm"の文字列）。
timeMin	@ 最も遅い有効な時刻（書式"hh:mm"の文字列）。
timeStep	@ ドロップダウンリストのエントリ間の分数。
timeFormat	@ 時刻のドロップダウンリスト内の値の表示に使用される書式文字列。
value	= 編集中の日付。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。
is-dropped-down-changing	& **isDroppedDownChanging** イベントハンドラ。
is-dropped-down-changed	& **isDroppedDownChanged** イベントハンドラ。
text-changed	& **textChanged** イベントハンドラ。
value-changed	& **valueChanged** イベントハンドラ。

WjInputMask クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

InputMask コントロール用のAngularJSディレクティブ。

wj-input-maskディレクティブは、AngularJSアプリケーションに**InputMask** コントロールを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>InputMaskコントロール:</p>
<wj-input-mask
  mask="99/99/99"
  mask-placeholder="*">
</wj-input-mask>
```

wj-input-maskディレクティブは以下の属性をサポートしています。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによってバインドされるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用してスコープにバインドされるコントロールプロパティを指定します。
control	= このディレクティブによって作成された**InputNumber** コントロールへの参照。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
mask	@ ユーザー入力時に値の書式設定に使用される文字列マスク。
prompt-char	@ マスク内の入力位置を示すために使用される文字。
place-holder	@ コントロールが空のときにヒントとして表示する文字列。
value	= 編集する文字列。
raw-value	= 編集する文字列（リテラル文字とプロンプト文字は含まない）。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。
value-changed	& **valueChanged** イベントハンドラ。

WjInputNumber クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

InputNumber コントロール用のAngularJSディレクティブ。

wj-input-numberディレクティブは、AngularJSアプリケーションに**InputNumber**コントロールを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>InputNumberコントロール:</p>
<wj-input-number
  value="theNumber"
  min="0"
  max="10"
  format="n0"
  placeholder="number between zero and ten">
</wj-input-number>
```

次の例では、**InputNumber**コントロールを作成し、異なる書式、範囲、ステップ値を使用した場合の効果を示します。

サンプル

Show me (https://jsfiddle.net/Wijmo5/u7HpD)

wj-input-numberディレクティブは以下の属性をサポートしています。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによってバインドされるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用してスコープにバインドされるコントロールプロパティを指定します。
control	= このディレクティブによって作成された**InputNumber** コントロールへの参照。
format	@ 数値の表示に使用される書式（**Globalize** を参照）。
input-type	@ コントロールによってホストされているHTML入力要素の"type"属性。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
max	@ 有効な最大の数値。
min	@ 有効な最小の数値。
place-holder	@ コントロールが空のときにヒントとして表示する文字列。
is-required	@ null値が禁止されるかどうかを示す値。
show-spinner	@ 値を**step**単位ずつ増減するスピナーボタンを表示するかどうかを示す値。
step	@ ユーザーがスピナーボタンをクリックしたときに値を増減する量。
text	= コントロールに表示するテキスト。
value	= 編集する数値。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。
text-changed	& **textChanged** イベントハンドラ。
value-changed	& **valueChanged** イベントハンドラ。

WjInputTime クラス

ファイル	wijmo.angular.js
モジュール	**wijmo.angular**
基本クラス	**WjComboBox**
表示	☐継承されたメンバー ☐イベント発生元

InputTime コントロール用のAngularJSディレクティブ。

wj-input-timeディレクティブを使用して、AngularJSアプリケーションに**InputTime**コントロールを追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>Here is an InputTime control:</p>
<wj-input-time
  value="theDate"
  format="h:mm tt"
  min="09:00" max="17:00"
  step="15">
</wj-input-time>
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/46PhD)

この例では、**InputDate** コントロールとInputTimeコントロールを使用して、 **Date**値（日時の情報を含む）を編集します。どちらのコントロールも同じコントローラ変数に連結されており、各コントロールはそれぞれの情報（日付または時刻）を編集します。この例では、 1 回のクリックで日付を選択できる **Calendar** コントロールも示しています。

wj-input-timeディレクティブは、次の属性を追加して**WjComboBox** を拡張します。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによって連結されるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用して、スコープに連結するコントロールプロパティを指定します。
control	= このディレクティブによって作成された**InputDate** コントロールへの参照。
format	@ 選択された時刻の表示に使用される書式。
mask	@ ユーザー入力時に入力の検証に使用されるマスク（**InputMask** を参照）。
max	@ 最も早い有効な時刻（書式"hh:mm"の文字列）。
min	@ 最も遅い有効な時刻（書式"hh:mm"の文字列）。
step	@ ドロップダウンリストのエントリ間の分数。
value	= 編集中の時刻を表すDateオブジェクト。
value-changed	& **valueChanged** イベントハンドラ。

min属性と**max**属性を指定する場合、それらは書式"hh:mm"の文字列にする必要があります。技術的には、W3Cの**[RFC 3339]**で定義されている任意の日時を使用できます。これは、標準HTML5入力要素でも使用される書式です。

WjItemTemplate クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

ListBox および**Menu** の項目テンプレート用のAngularJSディレクティブ。

wj-item-templateディレクティブは、**WjListBox** または**WjMenu** ディレクティブ内に記述する必要があります。

wj-item-templateディレクティブは、**ListBox**コントロールおよびデータバインドされた**Menu**コントロールの項目用のテンプレートを定義します。 このテンプレートには、任意のHTMLフラグメントとAngularJSバインディングおよびディレクティブを含めることができます。 コントローラーで使用可能なプロパティに加えて、ローカルの**$item**、**$itemIndex**、および**$control**テンプレート変数をAngularJSバインディングで使用してデータ項目、そのインデックス、およびオーナーコントロールを参照できます。

ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>項目テンプレートを持つListBoxコントロール:</p>
<wj-list-box items-source="musicians">
    <wj-item-template>
        {{$itemIndex}}.<b>{{$item.name}}</b>
        <br />
        <img ng-src="{{$item.photo}}"/>
    </wj-item-template>
</wj-list-box>
```

WjLinearGauge クラス

ファイル	wijmo.angular.js
モジュール	**wijmo.angular**
派生クラス	**WjBulletGraph**
表示	☐継承されたメンバー ☐イベント発生元

LinearGauge コントロール用のAngularJSディレクティブ。

wj-linear-gaugeディレクティブは、AngularJSアプリケーションに線形ゲージを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<wj-linear-gauge
  value="ctx.gauge.value"
  show-text="Value"
  is-read-only="false">
  <wj-range
    wj-property="pointer"
    thickness="0.2">
    <wj-range
      min="0"
      max="33"
      color="green">
    </wj-range>
    <wj-range
      min="33"
      max="66"
      color="yellow">
    </wj-range>
    <wj-range
      min="66"
      max="100"
      color="red">
    </wj-range>
  </wj-range>
</wj-linear-gauge>
```

wj-linear-gaugeディレクティブは以下の属性をサポートしています。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによってバインドされるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用してスコープにバインドされるコントロールプロパティを指定します。
control	= このディレクティブによって作成された**LinearGauge** コントロールへの参照。
direction	@ ゲージの値が増加する方向を示す**GaugeDirection** 値。
format	@ ゲージ値をテキストとして表示するために使用される書式文字列。
has-shadow	@ ゲージに影効果を付けるかどうかを示す値。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
is-animated	@ ゲージの値の変化をアニメーション表示するかどうかを示す値。
is-read-only	@ ユーザーが値を編集できないようにするかどうかを示す値。
min	@ ゲージに表示できる最小値。
max	@ ゲージに表示できる最大値。
show-text	@ どの値をゲージにテキストとして表示するかを示す **ShowText** 値。
step	@ ユーザーが方向キーを押したときにvalueプロパティを増減する量。
thickness	@ ゲージの太さ（0〜1のスケール）。
value	= ゲージに表示される値。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。

wj-linear-gaugeディレクティブには1つ以上の**WjRange** ディレクティブを含めることができます。

サンプル

Show me (https://jsfiddle.net/Wijmo5/t842jozb)

WjListBox クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

ListBox コントロール用のAngularJSディレクティブ。

wj-list-boxディレクティブは、AngularJSアプリケーションに**ListBox** コントロールを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
lt;p>ListBoxコントロール:</p>
<wj-list-box
  selected-item="theCountry"
  items-source="countries"
  placeholder="country">
</wj-list-box>
```

次の例では、**ListBox**コントロールを作成し、コントローラーによって公開された'countries'配列にコントロールをバインドします。選択した値は、**selected-item**属性を使用して'theCountry'コントローラープロパティにバインドされています。

サンプル

Show me (https://jsfiddle.net/Wijmo5/37GHw)

wj-list-boxディレクティブは以下の属性をサポートしています。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**selectedValue**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによってバインドされるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用してスコープにバインドされるコントロールプロパティを指定します。
control	= このディレクティブによって作成された**ListBox** コントロールへの参照。
display-member-path	@ 項目の視覚表示として使用するプロパティ。
is-content-html	@ 項目の内容がプレーンテキストとHTMLのどちらであるかを示す値。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
item-formatter	= リストに表示する値のカスタマイズに使用される関数。
items-source	= リスト項目を含む配列または**ICollectionView**。
max-height	@ リストの最大の高さ。
selected-index	= 現在選択されている項目のインデックス。
selected-item	= 現在選択されている項目。
selected-value	= **selected-value-path**を使用して取得された**selected-item**の値。
selected-value-path	@ **selected-item**から**selected-value**を取得するために使用されるプロパティ。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。
items-changed	& **itemsChanged** イベントハンドラ。
selected-index-changed	& **selectedIndexChanged** イベントハンドラ。

wj-list-boxディレクティブには、子ディレクティブとして**WjItemTemplate** を含めることができます。

WjMenu クラス

ファイル	wijmo.angular.js
モジュール	**wijmo.angular**
基本クラス	**WjComboBox**
表示	☐継承されたメンバー ☐イベント発生元

Menu コントロール用のAngularJSディレクティブ。

wj-menuディレクティブは、AngularJSアプリケーションにドロップダウンメニューを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>Menuコントロール:</p>
<wj-menu header="tax" value="tax">
  <wj-menu-item value="0">Exempt</wj-menu-item>
  <wj-menu-item value=".05">5%</wj-menu-item>
  <wj-menu-item value=".1">10%</wj-menu-item>
  <wj-menu-item value=".15">15%</wj-menu-item>
</wj-menu>
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/Wc5Mq)

この例では3つの**Menu**コントロールを作成します。最初のコントロールは値ピッカーとして使用します。2番目はパラメーターが付いたコマンドのリストを使用します。3番目は、コントローラーの**itemClicked**関数によって処理される3つのメニューのグループです。

wj-menuディレクティブは**WjComboBox** を拡張したもので、以下の属性を持ちます。

command-path	@ 項目がクリックされたときに実行されるコマンド。
command-parameter-path	@ コマンドのパラメーターを含むプロパティの名前。
header	@ コントロールに表示されるテキスト。
is-button	@ メニューのヘッダ領域をクリックしたときにメニューが反応するかどうか。
value	@ 選択された**wj-menu-item**のvalueプロパティの値。
item-clicked	& **itemClicked** イベントハンドラ。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。

wj-menuディレクティブには、子ディレクティブとして **WjMenuItem**、**WjMenuSeparator**、および**WjItemTemplate** （データバインドされたMenuコントロールの場合）を含めることができます。

WjMenuItem クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

メニュー項目用のAngularJSディレクティブ。

wj-menu-itemディレクティブは、**WjMenu** ディレクティブ内に記述する必要があります。 このディレクティブは以下の属性をサポートしています。

cmd = 項目がクリックされたときにコントローラーで実行する関数。
cmd-param = 項目がクリックされたときに**cmd**関数に渡されるパラメーター。
value = 項目がクリックされたときに選択する値（このプロパティと**cmd**のどちらか一方を使用します）。

項目によって表示される内容には、任意のHTMLフラグメントとAngularJSバインディングおよびディレクティブを含めることができます。 また、 **ng-repeat**ディレクティブと**ng-if**ディレクティブを使用してMenuコントロールの項目を設定することもできます。 コントローラーで使用可能なプロパティに加えて、ローカルの**$item**、**$itemIndex**、および**$control**テンプレート変数をAngularJSバインディングで使用してデータ項目、そのインデックス、およびオーナーコントロールを参照できます。

WjMenuSeparator クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

メニューセパレーター用のAngularJSディレクティブ。

wj-menu-item-separatorディレクティブは、**WjMenu** ディレクティブ内に記述する必要があります。 これは選択できないセパレーターをメニューに追加するもので、属性はありません。

WjMultiAutoComplete クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
基本クラス **WjAutoComplete**
表示 ☐継承されたメンバー ☐イベント発生元

MultiAutoComplete コントロール用のAngularJSディレクティブ。

wj-multi-auto-completeディレクティブを使用して、AngularJSアプリケーションに**MultiAutoComplete**コントロールを追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>オブジェクトのコレクションに連結されたInputTagsコントロールの例</p>
<wj-multi-auto-complete
    max-selected-items="8"
    items-source="ctx.items"
    selected-Member-path="selected">
</wj-multi-auto-complete>
```

wj-multi-auto-completeディレクティブは、次の属性で**WjAutoComplete** を拡張します。

max-selected-items @ 選択できる最大項目数。
selected-member-path @ どの項目が選択されるかの制御に使用されるプロパティの名前。
selected-items @ 現在選択されている項目を含む配列。
selected-items-changed & **selectedItemsChanged** イベントハンドラ。

WjMultiRow クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
基本クラス **WjFlexGrid**
表示 ☐継承されたメンバー ☐イベント発生元

MultiRow コントロール用のAngularJSディレクティブ。

wj-multi-rowディレクティブを使用して、AngularJSアプリケーションに**MultiRow**コントロールを追加します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。

wj-multi-rowディレクティブは、次の属性で**WjFlexGrid** を拡張します。

control = このディレクティブによって作成された**MultiRow** コントロールへの参照。
layout-definition @ 各データ項目の表示に使用される行のレイアウトを定義する値。
collapsed-headers @列ヘッダーを折りたたみ、単一の行としてグループヘッダーを表示する かどうかを判定する値を取得または設定します。
center-headers-vertically @ 複数の行にまたがるセルのコンテンツを垂直方向に中央揃えにするかどうかを判定する値を取得または設定します。
show-header-collapse-button @ ユーザーが列ヘッダーの折りたたみと展開を行うためのボタンを グリッドの列ヘッダーパネルに表示するかどうかを判定する値を取得または設定します。

WjMultiSelect クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
基本クラス **WjComboBox**
表示 ☐継承されたメンバー ☐イベント発生元

MultiSelect コントロール用のAngularJSディレクティブ。

wj-multi-selectディレクティブは、AngularJSアプリケーションに**MultiSelect**コントロールを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>オブジェクトのコレクションにバインドされたMultiSelectコントロール:</p>
<wj-multi-select
    placeholder="Select Countries"
    items-source="ctx.items"
    header-format="{count} countries selected"
    display-Member-path="country"
    checked-Member-path="selected">
</wj-multi-select>
```

wj-multi-selectディレクティブは**WjComboBox** を拡張したもので、以下の属性を持ちます。

checked-member-path @ 各項目の横に配置されたチェックボックスを制御するために使用されるプロパティの名前。
header-format @ コントロールで**maxHeaderItems**の数を超える項目がチェックされているときにヘッダの内容の作成に使用される書式文字列。
header-formatter = コントロールのヘッダのHTMLを取得する関数。
max-header-items @ コントロールのヘッダに表示する項目の最大数。
checked-items-changed & **checkedItemsChanged** イベントハンドラ。

WjPdfViewer クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

PdfViewer コントロール用のAngularJSディレクティブ。

wj-pdf-viewerディレクティブを使用して、AngularJSアプリケーションにPDFビューアを追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<wj-pdf-viewer  paginated="false"
  service-url="ctx.serviceUrl"
  file-path="ctx.path"
  report-name="ctx.reportName">
</wj-pdf-viewer;
```

wj-pdf-viewerディレクティブは、次の属性をサポートします。

service-url	@ C1 Web APIサービスのアドレスを示す値。
file-path	@ サーバー上のドキュメントへの完全パスを示す値。
control	= このディレクティブによって作成された**PdfViewer** コントロールへの参照。
full-screen	@ ビューアが全画面表示モードかどうかを示す値。
zoom-factor	@ ドキュメントページを表示する現在のズーム倍率を示す値。
mouse-mode	@ ビューパネルでマウスをクリックアンドドラッグして テキストを選択するかどうかを示す値。
initialized	& このイベントは、属性値でコントロールを初期化して 連結処理が完了した後に発生します。
is-initialized	= 属性値でコントロールを初期化して 連結処理が完了したかどうかを示す値。
view-mode	@ ドキュメントページを表示する方法を示す値。
request-headers	@ データを送信または要求するときに使用されるリクエストヘッダーを含むオブジェクト。
page-index-changed	& **pageIndexChanged** イベントのハンドラ。
view-mode-changed	& **viewModeChanged** イベントのハンドラ。
mouse-mode-changed	& **mouseModeChanged** イベントのハンドラ。
full-screen-changed	& **fullScreenChanged** イベントのハンドラ。
zoom-factor-changed	& **zoomFactorChanged** イベントのハンドラ。
query-loading-data	& **queryLoadingData** イベントのハンドラ。
before-send-request	& **beforeSendRequest** イベントのハンドラ。

WjPivotChart クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

PivotChart コントロール用のAngularJSディレクティブ。

wj-pivot-chartディレクティブと**wj-pivot-panel**ディレクティブを使用して、AngularJSアプリケーションにピボットチャートを追加できます。

ディレクティブ名とパラメータ名は、キャメルケースではなく、小文字とダッシュの形式にする必要があります。次に例を示します。

```
<wj-pivot-panel
    control="thePanel"
    items-source="rawData">
</wj-pivot-panel>
<wj-pivot-chart
    items-source="thePanel"
    chart-type="Bar"
    max-series="10"
    max-points="100">
</wj-pivot-chart>
```

wj-pivot-chartディレクティブは、次の属性をサポートします。

items-source この**PivotChart** によって表示されるビューを定義する**PivotPanel** を取得または設定します。
chart-type 表示するチャートのタイプを定義する**PivotChartType** 値を取得または設定します。
show-hierarchical-axes グループ化されたデータに対してチャートの軸注釈をグループ化するかどうかを取得または設定します。
stacking 系列オブジェクトを積み重ねるかどうかを決定する**Stacking** 値と、積み重ねる場合はその方法を取得または設定します。
show-totals チャートに合計だけを含めるかどうかを取得または設定します。
max-series チャートに表示するデータ系列の最大数を取得または設定します。
max-points 各系列に表示するポイントの最大数を取得または設定します。

WjPivotGrid クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
基本クラス **WjFlexGrid**
表示 ☐継承されたメンバー ☐イベント発生元

PivotGrid コントロール用のAngularJSディレクティブ。

wj-pivot-gridディレクティブと**wj-pivot-panel**ディレクティブを使用して、AngularJSアプリケーションにピボットテーブルを追加できます。

ディレクティブ名とパラメータ名は、キャメルケースではなく、小文字とダッシュの形式にする必要があります。次に例を示します。

```
<wj-pivot-panel
    control="thePanel"
    items-source="rawData">
</wj-pivot-panel>
<wj-pivot-grid
    items-source="thePanel"
    show-detail-on-double-click="false"
    custom-context-menu="true">
</wj-pivot-grid>
```

wj-pivot-gridディレクティブは**wj-flex-grid**ディレクティブを拡張し、次の属性のサポートが追加されています。

items-source	この**PivotGrid** によって表示されるビューを定義する**PivotPanel** を取得または設定します。
show-detail-on-double-click	ユーザーがセルをダブルクリックしたときに、グリッドが詳細レコードを含むポップアップを表示するかどうかを取得または設定します。
custom-context-menu	グリッドが、フィールド設定を変更したり詳細レコードを表示するためのコマンドを含むカスタムコンテキストメニューを提供するかどうかを取得または設定します。
collapsible-subtotals	ユーザーがグリッドで行および列の小計グループの折りたたみと展開を行えるかどうかを取得または設定します。
center-headers-vertically	ヘッダーセルのコンテンツを垂直方向に中央揃えにするかどうかを取得または設定します。

WjPivotPanel クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

PivotPanel コントロール用のAngularJSディレクティブ。

wj-pivot-panelディレクティブを**wj-pivot-grid**ディレクティブと**wj-pivot-chart**ディレクティブのデータソースとして使用して、AngularJSアプリケーションにピボットテーブルとピボットチャートを追加できます。

ディレクティブ名とパラメータ名は、キャメルケースではなく、小文字とダッシュの形式にする必要があります。次に例を示します。

```
<wj-pivot-panel
    control="thePanel"
    items-source="rawData">
</wj-pivot-panel>
<wj-pivot-grid
    items-source="thePanel"
    show-detail-on-double-click="false"
    custom-context-menu="true">
</wj-pivot-grid>
```

wj-pivot-panelディレクティブは、次の属性をサポートします。

items-source	ピボットビューを生成するために使用される生データを取得または設定します。
auto-generate-fields	**itemsSource** に基づいて、パネルにフィールドコレクションを自動的に挿入するかどうかを取得または設定します。
view-definition	現在のピボットビュー定義をJSON文字列として取得または設定します。
engine	データの集約を行う**PivotEngine** への参照を取得します。

WjPopup クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

Popup コントロール用のAngularJSディレクティブ。

wj-popupディレクティブは、AngularJSアプリケーションに**Popup** コントロールを追加する場合に使用します。

ポップアップの内容は**wj-popup**タグ内で指定します。これには、任意のHTMLフラグメントとAngularJSバインディングおよびディレクティブを含めることができます。

ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<p>ボタンによって表示されるPopupコントロール:</p>
<button id="btn2" type="button">
    Click to show Popup
</button>
<wj-popup owner="#btn2" show-trigger="Click" hide-trigger="Blur">
    <h3>
        Salutation
    </h3>
    <div class="popover-content">
        Hello {{firstName}} {{lastName}}
    </div>
</wj-popup>
```

wj-popupディレクティブは以下の属性をサポートしています。

control = このディレクティブによって作成されたPopupコントロールへの参照。
fade-in @ フェードインアニメーションを使用してポップアップを表示するかどうかを決定するブール値。
fade-out @ フェードアウトアニメーションを使用してポップアップを非表示にするかどうかを決定するブール値。
hide-trigger @ **Popup** を非表示にするアクションを定義する**PopupTrigger** 値。
initialized & このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized = バインディングによるコントロールの初期化が完了したかどうかを示す値。
owner @ ポップアップの表示/非表示を制御する要素を参照するCSSセレクター。
show-trigger @ **Popup** を表示するアクションを定義する**PopupTrigger** 値。
modal @ **Popup** をモーダルダイアログとして表示するかどうかを決定するブール値。
got-focus & **gotFocus** イベントハンドラ。
lost-focus & **lostFocus** イベントハンドラ。
showing & **showing** イベントハンドラ。
shown & **shown** イベントハンドラ。
hiding & **hiding** イベントハンドラ。
hidden & **hidden** イベントハンドラ。

WjRadialGauge クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

RadialGauge コントロール用のAngularJSディレクティブ。

wj-radial-gaugeディレクティブは、AngularJSアプリケーションに放射状ゲージを追加する場合に使用します。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<b>RadialGauge</b>コントロール:</p>
<wj-radial-gauge
  style="height:300px"
  value="count"
  min="0" max="10"
  is-read-only="false">
</wj-radial-gauge>
```

wj-radial-gaugeディレクティブは以下の属性をサポートしています。

ng-model	@ Angularのng-modelディレクティブを使用してコントロールの**value**プロパティをバインドします。ng-modelディレクティブを使用してプロパティをバインドすると、データ検証やコントロールの状態のフォームインスタンスへの公開など、ng-modelの持つ利点を活用できます。ng-modelディレクティブによってバインドされるコントロールのプロパティを再定義するには、wj-model-property属性を使用します。
wj-model-property	@ **ng-model**ディレクティブを使用してスコープにバインドされるコントロールプロパティを指定します。
control	= このディレクティブによって作成されたRadialGaugeコントロールへの参照。
auto-scale	@ ゲージをホスト要素いっぱいに拡大するかどうかを示す値。
format	@ ゲージ値をテキストとして表示するために使用される書式文字列。
has-shadow	@ ゲージに影効果を付けるかどうかを示す値。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
is-animated	@ ゲージの値の変化をアニメーション表示するかどうかを示す値。
is-read-only	@ ユーザーが値を編集できないようにするかどうかを示す値。
min	@ ゲージに表示できる最小値。
max	@ ゲージに表示できる最大値。
show-text	@ どの値をゲージにテキストとして表示するかを示す **ShowText** 値。
step	@ ユーザーが方向キーを押したときにvalueプロパティを増減する量。
start-angle	@ ゲージの開始角度（度単位、9時の位置から時計回りに測定）。
sweep-angle	@ ゲージの掃引角度（度単位、正または負の数値を指定可能）。
thickness	@ ゲージの太さ（0〜1のスケール）。
value	= ゲージに表示される値。
got-focus	& **gotFocus** イベントハンドラ。
lost-focus	& **lostFocus** イベントハンドラ。

wj-radial-gaugeディレクティブには1つ以上の**WjRange** ディレクティブを含めることができます。

サンプル

Show me (https://jsfiddle.net/Wijmo5/7ec2144u)

WjRange クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

Range オブジェクト用のAngularJSディレクティブ。

wj-rangeディレクティブは、**WjLinearGauge**、**WjRadialGauge**、または**WjBulletGraph** ディレクティブ内に記述する必要があります。これは親ディレクティブの'ranges'配列にRangeオブジェクトを追加します。また、wj-property属性にプロパティ名を指定することで、親ディレクティブのその他のRange型プロパティを初期化することもできます。

次に例を示します。

```
<wj-radial-gauge
    min="0"
    max="200"
    step="20"
    value="theValue"
    is-read-only="false">
    <wj-range
      min="0"
      max="100"
      color="red">
    </wj-range>
    <wj-range
      min="100"
      max="200"
      color="green">
    </wj-range>
    <wj-range
      wj-property="pointer"
      color="blue">
    </wj-range>
</wj-radial-gauge>
```

wj-rangeディレクティブは以下の属性をサポートしています。

min @ 範囲の最小値。
max @ 範囲の最大値。
color @ 範囲の表示に使用される色。
thickness @ 範囲の太さ（0〜1のスケール）。
name @ 範囲の名前。
wj-property @ このディレクティブによって初期化するプロパティの名前。

WjReportViewer クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

ReportViewer コントロール用のAngularJSディレクティブ。

wj-report-viewerディレクティブを使用して、AngularJSアプリケーションにレポートビューアを追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<wj-report-viewer paginated="true"
  service-url="ctx.serviceUrl"
  file-path="ctx.path"
  report-name="ctx.reportName">
</wj-report-viewer;
```

wj-report-viewerディレクティブは、次の属性をサポートします。

service-url	@ C1 Web APIサービスのアドレスを示す値。
file-path	@ サーバー上のドキュメントへの完全パスを示す値。
report-name	@ FlexReportの場合は、 FlexReport定義ファイルに定義されたレポート名でこれを設定します。 SSRSレポートの場合は、これを空の文字列のままにしてください。
control	= このディレクティブによって作成された**ReportViewer** コントロールへの参照。
full-screen	@ ビューアが全画面表示モードかどうかを示す値。
zoom-factor	@ ドキュメントページを表示する現在のズーム倍率を示す値。
mouse-mode	@ ビューパネルでマウスをクリックアンドドラッグして テキストを選択するかどうかを示す値。
initialized	& このイベントは、属性値でコントロールを初期化して 連結処理が完了した後に発生します。
is-initialized	= 属性値でコントロールを初期化して 連結処理が完了したかどうかを示す値。
view-mode	@ ドキュメントページを表示する方法を示す値。
paginated	@ コンテンツを一連の固定サイズのページとして 表すかどうかを示す値。
parameters	@ A dictionary of {name: value} pairs that describe the parameters used to run the report.
request-headers	@ データを送信または要求するときに使用されるリクエストヘッダーを含むオブジェクト。
page-index-changed	& **pageIndexChanged** イベントのハンドラ。
view-mode-changed	& **viewModeChanged** イベントのハンドラ。
mouse-mode-changed	& **mouseModeChanged** イベントのハンドラ。
full-screen-changed	& **fullScreenChanged** イベントのハンドラ。
zoom-factor-changed	& **zoomFactorChanged** イベントのハンドラ。
query-loading-data	& **queryLoadingData** イベントのハンドラ。
before-send-request	& **beforeSendRequest** イベントのハンドラ。

WjSheet クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

Sheet オブジェクト用のAngularJSディレクティブ。

wj-sheetディレクティブは、**WjFlexSheet** ディレクティブ内に記述する必要があります。 次の属性をサポートします。

name @ シートの名前。
row-count @ バインドされていないシートの行数の初期設定。 AngularJSによって**Sheet** が初期化された後にこの属性を変更しても、効果はありません。
column-count @ バインドされていないシートの列数の初期設定。 AngularJSによって**Sheet** が初期化された後にこの属性を変更しても、効果はありません。
items-source = データバインドされたシートのデータソース。 AngularJSによって**Sheet** が初期化された後にこの属性を変更しても、効果はありません。
visible @ シートが表示されるかどうかを示す値。
name-changed & **nameChanged** イベントハンドラ。

WjSlicer クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

Slicer コントロール用のAngularJSディレクティブ。

wj-slicer ディレクティブを使用することで**PivotField** オブジェクトに適用されたフィルタをすぐに編集できます。

ディレクティブ名とパラメータ名は、キャメルケースではなく、小文字とダッシュの形式にする必要があります。次に例を示します。

```
<wj-slicer
    field="theField"
    header="theHeader"
    show-header="true">
</wj-slicer>
```

The **wj-slicer**ディレクティブは、次の属性をサポートします。

field **Slicer** によってフィルタされたPivotFieldを取得または設定します。
header **Slicer** の上部に表示されるヘッダ文字列を取得または設定します。
showHeader ヘッダ文字列とマルチセレクトやクリアボタンを持つヘッダ部分を表示するかどうかを示す値を取得または設定します。
showCheckboxes 各項目に対してチェックボックスを表示するかどうかを示す値を取得または設定します。
multiSelect ユーザーがリストから複数の項目を選択できるかどうかを示す値を取得または設定します。

WjSunburst クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
基本クラス **WjFlexPie**
表示 ☐継承されたメンバー ☐イベント発生元

Sunburst コントロール用のAngularJSディレクティブ。

child-items-path = 階層化データの子項目の生成に使用される配列または文字列オブジェクト。

WjTabPanel クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

TabPanel コントロール用のAngularJSディレクティブ。

wj-tab-panel ディレクティブを使用して、 AngularJSアプリケーションにTabPanelを追加できます。 ディレクティブ名とパラメータ名は、キャメルケースではなく、小文字とダッシュの形式にする必要があります。 次に例を示します。

```
<wj-tab-panel>
   <wj-tab>
     <a>Tab1ヘッダー</a>
     <div>
         Tab1の内容
     </div>
   </wj-tab>
   <wj-tab is-disabled="true">
     <a>Tab2ヘッダー</a>
     <div>
         Tab2の内容
     </div>
   </wj-tab>
</wj-tab-panel>
```

wj-tab-panel ディレクティブは以下の属性をサポートしています。

is-animated	@ タブの切り替えをフェードイン効果でアニメーション化するかどうかを決定する値。
auto-switch	@ ユーザーが方向キーを使用してタブを選択したときにコントロールがタブを自動的に切り替えるかどうかを決定する値。
control	= このディレクティブによって作成された**TabPanel** コントロールへの参照。
selected-index	= 現在選択されている（アクティブな）タブのインデックス。
selected-tab	= 現在選択されている**Tab** オブジェクト。
initialized	& このイベントは、バインディングが属性値でコントロールを初期化した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が属性値を含めて完了したかどうかを示す値。
selected-index-changed	& **selectedIndexChanged** イベントハンドラ。

wj-tab-panel ディレクティブには1つ以上の **WjTab** ディレクティブを含めることができます。

WjTooltip クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

Tooltip クラス用のAngularJSディレクティブ。

wj-tooltipディレクティブは、ページ上の要素にツールチップを追加する場合に使用します。 wj-tooltipディレクティブはHTMLコンテンツ、スマート配置、およびタッチをサポートします。

wj-tooltipディレクティブは、ツールチップを適用する対象の要素に対する追加パラメーターとして指定します。パラメーターの値は、ツールチップのテキストまたはテキストを含む要素のIDです。次に例を示します。

```
<p wj-tooltip="#fineprint" >
    通常の段落の内容...</p>
<div id="fineprint" style="display:none">
  <h3>重要</h3>
  <p>
    今四半期のデータは比例配分によって得られた推定値です。</p>
</div>
```

WjTreeMap クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

TreeMap コントロール用のAngularJSディレクティブ。

items-source	= チャートのデータを含む配列または**ICollectionView** オブジェクト。
binding	@ 項目の値を含むプロパティの名前。
binding-name	@ 項目の名前を含むプロパティの名前。この名前は配列または文字列である必要があります。
footer	@ チャートのフッタに表示するテキスト（プレーンテキスト）。
footer-style	= チャートのフッタに適用するスタイル。
header	@ チャートのヘッダに表示するテキスト（プレーンテキスト）。
header-style	= チャートのヘッダに適用するスタイル。
initialized	& このイベントは、バインディングによるコントロールの初期化が完了した後に発生します。
is-initialized	= バインディングによるコントロールの初期化が完了したかどうかを示す値。
type	@ TreeMapのタイプ。
child-items-path	@ 階層データ内の子アイテムを生成するために使用される1つ以上のプロパティの名前を示す値。
max-depth	= 現在のビューに表示するノードレベルの最大数。
palette	= TreeMap項目の表示に使用されるデフォルトの色を含む配列。
plot-margin	= コントロールの端からプロット領域端までの間隔のピクセル数、またはCSSスタイルのマージン。
tooltip-content	@ **ChartTooltip** のcontentプロパティに表示する値。
label-content	@ **DataLabel** のcontentプロパティに表示する値。
rendering	& **rendering** イベントハンドラ。
rendered	& **rendered** イベントハンドラ。

wj-tree-map ディレクティブには、子ディレクティブとして**WjFlexChartLegend** および**WjFlexChartDataLabel** を含めることができます。

WjTreeView クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

TreeView コントロール用のAngularJSディレクティブ。

wj-tree-viewディレクティブを使用して、AngularJSアプリケーションにTreeViewを追加できます。 ディレクティブ名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。次に例を示します。

```
<wj-tree-view  items-source="items"
  display-member-path="ctx.displayMemberPath"
  child-items-path="ctx.childItemsPath"
  is-animated="ctx.isAnimated">
<wj-tree-view;
```

wj-tree-viewディレクティブは、次の属性をサポートします。

items-source	=	**TreeView** 項目を含む配列。
child-items-path	@	各ノードの子項目を含む1つ以上のプロパティの名前を示す値。
control	=	このディレクティブによって作成された**TreeView** コントロールへの参照。
display-member-path	@	ノードのビジュアル表現として使用する1つ以上のプロパティの名前を示す値。
image-member-path	@	ノードの画像のソースとして使用する1つ以上のプロパティの名前を示す値。
is-content-html	@	項目がプレーンテキストとHTMLの どちらに連結されているかを示す値。
initialized	&	このイベントは、連結が属性値によるコントロールの初期化を 完了した後に発生します。
is-initialized	=	連結が、属性値によるコントロールの初期化を 完了したかどうかを示す値。
show-checkboxes	@	**TreeView** で、ノードにチェックボックスを追加し、 その状態を管理するかどうかを指定する値。
auto-collapse	@	ノードが展開されたときに、兄弟ノードを折りたたむかどうかを指定する値。
is-animated	@	ノードを展開または折りたたむときにアニメーションを使用するかどうかを示す値。
is-readOnly	@	ユーザーがノード内のテキストを編集できるかどうかを指定する値。
allow-dragging	@	ユーザーが**TreeView** 内でノードをドラッグアンドドロップできるかどうかを指定する値。
expand-on-click	@	ユーザーがノードヘッダーをクリックしたとき、に折りたたまれているノードを展開するかどうかを指定する値。
selected-item	@	現在選択されているデータ項目を示す値。
selected-node	@	現在選択されている**TreeNode** を示す値。
checked-items	@	現在オンに設定されている項目を含む配列。
lazy-load-function	=	オンデマンドで子ノードをロードする関数。
items-source-changed	&	**itemsSourceChanged** イベントハンドラ。
loading-items	&	**loadingItems** イベントハンドラ。
loaded-items	&	**loadedItems** イベントハンドラ。
item-clicked	&	**itemClicked** イベントハンドラ。
selected-item-changed	&	**selectedItemChanged** イベントハンドラ。
checked-items-Changed	&	**checkedItemsChanged** イベントハンドラ。
is-collapsed-changing	&	**isCollapsedChanging** イベントハンドラ。
is-collapsed-changed	&	**isCollapsedChanged** イベントハンドラ。
is-checked-changing	&	**isCheckedChanging** イベントハンドラ。
is-checked-changed	&	**isCheckedChanged** イベントハンドラ。
format-item	&	**formatItem** イベントハンドラ。
drag-start	&	**dragStart** イベントハンドラ。
drag-over	&	**dragOver** イベントハンドラ。
drop	&	**drop** イベントハンドラ。
drag-end	&	**dragEnd** イベントハンドラ。
node-edit-starting	&	**nodeEditStarting** イベントハンドラ。
node-edit-started	&	**nodeEditStarted** イベントハンドラ。
node-edit-ending	&	**nodeEditEnding** イベントハンドラ。
node-edit-ended	&	**nodeEditEnded** イベントハンドラ。

WjValidationError クラス

ファイル wijmo.angular.js
モジュール **wijmo.angular**
表示 ☐継承されたメンバー ☐イベント発生元

式に基づくカスタム検証用のAngularJSディレクティブ。

wj-validation-errorディレクティブは、AngularJSとHTML5ネイティブの両方の 検証メカニズムをサポートしています。このディレクティブはAngularJS式を受け取ります。無効な入力が行われたり、入力が有効でも 空の文字列である場合に、その式からエラーメッセージ文字列を返すように設定します。

AngularJS検証の場合は、**ng-model**ディレクティブと組み合わせて使用する必要があります。 その場合、**wj-validation-error**ディレクティブは、**wjValidationError**エラーキーを指定した**NgModelController.$setValidity**メソッドの呼び出しを使用してエラーを報告します。 これは、AngularJSネイティブおよびカスタム検証ディレクティブの動作と同じです。

HTML5検証の場合、**wj-validation-error**ディレクティブは、 HTML5検証API**setCustomValidity**メソッドを使用して、要素にエラー状態を設定します。次に例を示します。

```
<p>HTML5検証</p>
<form>
    <input type="password"
        placeholder="Password"
        name="pwd" ng-model="thePwd"
        required minlength="2" />
    <input type="password"
        placeholder="Check Password"
        name="chk" ng-model="chkPwd"
        wj-validation-error="chkPwd != thePwd ?'Passwords don\'t match' :''" />
</form>

<p>AngularJS検証</p>
<form name="ngForm" novalidate>
    <input type="password"
        placeholder="Password"
        name="pwd" ng-model="thePwd"
        required minlength="2" />
    <input type="password"
        placeholder="Check Password"
        name="chk" ng-model="chkPwd"
        wj-validation-error="chkPwd != thePwd" />
    <div
        ng-show="ngForm.chk.$error.wjValidationError && !ngForm.chk.$pristine">
        Sorry, the passwords don't match.
    </div>
</form>
```

CellTemplateType 列挙体

ファイル `wijmo.angular.js`
モジュール **wijmo.angular**

テンプレートを適用するセルのタイプを定義します。 この値は、**WjFlexGridCellTemplate** ディレクティブの**cell-type**属性で指定します。

メンバー

名前	値	説明
Cell	0	通常の（データ）セルを定義します。
CellEdit	1	編集モードのセルを定義します。
ColumnHeader	2	列ヘッダセルを定義します。
RowHeader	3	行ヘッダセルを定義します。
RowHeaderEdit	4	編集モードの行ヘッダセルを定義します。
TopLeft	5	左上のセルを定義します。
GroupHeader	6	グループ行のグループヘッダセルを定義します。
Group	7	グループ行の通常のセルを定義します。
ColumnFooter	8	列フッターセルを定義します。
BottomLeft	9	左下のセル（行ヘッダーと列フッターが交差する位置にあるセル）を定義します。

wijmo.knockout モジュール

ファイル `wijmo.knockout.js`
モジュール **wijmo.knockout**

Wijmoコントロール用のKnockoutJSバインディングが含まれます。

ページのマークアップ中の特定の場所にWijmoコントロールを追加するには、** **要素を追加し、**data-bind**属性でコントロールのバインディングを定義します。バインディング名は、wjが前に付いたコントロール名に対応します。たとえば、**wjInputNumber** バインディングはWijmo **InputNumber** コントロールを表します。

ページのマークアップ中の特定の場所にWijmoコントロールを追加するには、** **要素を追加し、**data-bind**属性でコントロールのバインディングを定義します。 バインディング名は、wjが前に付いたコントロール名に対応します。たとえば、**wjInputNumber** バインディングはWijmo **InputNumber** コントロールを表します。バインディング値は、コントロールの読み取り/書き込みプロパティ名およびイベント名に対応するプロパティと、それらに対応するコントロールプロパティの値およびイベントハンドラを定義する値のペアを含むオブジェクトリテラルです。

Wijmo **InputNumber**コントロールを作成する例を次に示します。この例では、InputNumberコントロールの**value**プロパティをビューモデルの**theValue**プロパティにバインドし、**step**プロパティを1に設定し、**inputType**プロパティを'text'に設定しています。

```
<div data-bind="wjInputNumber:{ value: theValue, step:1, inputType:'text' }"></div>
```

カスタム要素

Wijmo for Knockoutでは、標準のKnockoutバインディング構文を使用する代わりに、ページマークアップでWijmoコントロールをカスタム要素として宣言することもできます。カスタム要素では、タグ名がコントロールのバインディング名に対応し、属性名がコントロールのプロパティ名に対応します。要素名とパラメーター名の書式は、キャメルケースではなくダッシュで結んだ小文字にする必要があります。上記の例で示したコントロールをカスタム要素構文で定義すると、次のようになります。

```
<wj-input-number value="theValue" step="1" input-type="'text'"></wj-input-number>
```

注意すべきことは、属性値を定義する際にdata-bind定義の場合とまったく同じJavaScript式構文を使用することです。このような要素は、Wijmo for Knockoutプリプロセッサによって従来のdata-bind形式に変換されます。詳細については、「**カスタム要素のプリプロセッサ**」トピックを参照してください。

コントロールのプロパティへのバインド

WijmoのKnockoutJSバインディングは、コントロールの任意の読み取り/書き込みプロパティへのバインドをサポートします。任意の有効なKnockoutJS式（たとえば、定数、ビューモデルのオブザーバブルプロパティ、複合式など）をプロパティに割り当てることができます。

Note that binding expression should resolve (after calling **ko.unwrap(expression)** on it) to a pure JavaScript value understandable by the corresponding Wijmo JavaScript control. This in particular means that you can't bind the **itemsSource** property of Wijmo controls to a Knockout **observableArray**, or to array whose items' properties are Knockout **observable**(s).

ほとんどのプロパティは一方向バインディングを提供します。つまり、バインドされたオブザーバブルなビューモデルプロパティが変更されるとそのオブザーバブルのバインド先のコントロールプロパティも変更されますが、その逆は起こりません。 ただし、一部のプロパティは双方向バインディングをサポートします。つまり、コントロールプロパティに加えられた変更が、そのコントロールプロパティにバインドされているオブザーバブルにも伝播します。双方向バインディングは、コントロール自体、コントロールに対するユーザーの操作、またはその他の事象によって変更可能なプロパティで使用されます。たとえば、InputNumberコントロールの**value**プロパティと**text**プロパティは双方向バインディングをサポートします。これらのプロパティは、ユーザーが新しい値を入力しているときにコントロールによって変更されます。 InputNumberの残りのプロパティは一方向バインディングモードで動作します。

コントロールのイベントへのバインド

コントロールのイベントにハンドラをアタッチするには、コントロールのバインディングを定義するオブジェクトリテラルのプロパティとしてイベント名を定義し、このイベントで呼び出す関数をプロパティの値として指定します。 Wijmoバインディングは、イベントハンドラの定義に関して、 **click**や**event**といったKnockoutJSの組み込みバインディングで使用されるルールと同じルールに従います。イベントハンドラは以下のパラメーターのセットを以下の順序で受け取ります。

- **data:** 現在のモデルの値。**click**や**event**といったKnockoutJSの組み込みバインディングの場合と同じです。
- **sender:** イベントの送信元。
- **args:** イベント引数。

以下の例では、**InputNumber**コントロールを作成し、**valueChanged**イベントのイベントハンドラとしてコントロールの新しい値をダイアログで表示する関数を指定しています。

```
<!-- HTML -->
<div data-bind="wjInputNumber:{ value: theValue, step:1, valueChanged: valueChangedEH }"></div>


// ビューモデル
this.valueChangedEH = function (data, sender, args) {
    alert('The new value is:' + sender.value);
}
```

同じコントロールをカスタム要素構文で定義すると、次のようになります。

```
<wj-input-number value="theValue" step="1" value-changed="valueChangedEH"></wj-input-number>
```

undefined値を持つオブザーバブルへのバインド

*undefined*値が割り当てられたビューモデルのオブザーバブルプロパティには、初期化フェーズでWijmoバインディングによって特別な処理がなされます。たとえば、ko.observable(undefined)またはko.observable()として作成したオブザーバブルをコントロールのプロパティにバインドした場合、Wijmoはそのコントロールに値を割り当てません。双方向バインディングをサポートするプロパティでは、バインドされたオブザーバブルは初期化後にコントロールのプロパティの値で更新されるため、これはコントロールのデフォルト値でオブザーバブルを初期化する手段となります。 *null*値を持つオブザーバブル（ko.observable(null)など）には通常の処理がなされ、バインド先のコントロールプロパティにnullが割り当てられる点に注意してください。最初の初期化が完了した後は、undefined値を持つオブザーバブルにもWijmoの通常の処理がなされるようになり、コントロールのプロパティにundefinedが割り当てられます。

以下の例では、**InputNumber**コントロールの**value**プロパティは初期化後にデフォルト値の0となり、その同じ値がビューモデルの**theValue**プロパティにも割り当てられます。

```
<!-- HTML -->
<div data-bind="wjInputNumber:{ value: theValue }"></div>


// ビューモデル
this.theValue = ko.observable();
```

複合プロパティおよび配列プロパティの定義

一部のWijmoコントロールは、配列または複合オブジェクトを含むプロパティを持ちます。たとえば、**FlexChart** コントロールの**axisX**プロパティと**axisY**プロパティは**Axis** オブジェクトを表し、**series**プロパティは**Series** オブジェクトの配列です。Wijmoには、コントロール要素の子要素に追加するこのような型について特別なバインディングが用意されています。コントロールが同じ複合型のプロパティを複数公開している場合は、複合型バインディングの**wjProperty**プロパティによって、定義するコントロールプロパティを指定します。

以下の例では、**FlexChart**を作成し、**axisX**および**axisY**プロパティと2つの系列オブジェクトを定義しています。

```
<div data-bind="wjInputNumber:{ itemsSource: data, bindingX:'country' }">
    <div data-bind="wjFlexChartAxis:{ wjProperty:'axisX', title: chartProps.titleX }"></div>
    <div data-bind="wjFlexChartAxis:{ wjProperty:'axisY', title: chartProps.titleY }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Sales', binding:'sales' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Downloads', binding:'downloads' }"></div>
</div>
```

同じコントロールをカスタム要素構文で定義すると、次のようになります。

```
<wj-flex-chart items-source="data" binding-x="'country'">
    <wj-flex-chart-axis wj-property="'axisX'" title="chartProps.titleX"></wj-flex-chart-axis>
    <wj-flex-chart-axis wj-property="'axisY'" title="chartProps.titleY"></wj-flex-chart-axis>
    <wj-flex-chart-series name="'Sales'" binding"'sales'"></wj-flex-chart-series>
    <wj-flex-chart-series name="'Downloads'" binding"'downloads'"></wj-flex-chart-series>
</wj-flex-chart>
```

controlプロパティ

各Wijmoコントロールバインディングは、そのバインディングによって作成されたWijmoコントロールインスタンスを参照する **control**プロパティを公開しています。これを使用して、ビューモデルのコードや他のバインディングでコントロールを参照できます。 たとえば、以下の例では、**FlexGrid** コントロールを作成してその参照をビューモデルの **flex**オブザーバブルプロパティに格納し、それをボタンクリックイベントハンドラで使用してボタンのクリック時にグリッドの次のレコードに移動するようにしています。

```
<!-- HTML -->
<div data-bind="'wjFlexGrid':{ itemsSource: data, control: flex }"></div>
<button data-bind="click: moveToNext">Next</button>


// ビューモデル
this.flex = ko.observable();
this.moveToNext = function () {
    this.flex().collectionView.moveCurrentToNext();
}
```

initializedイベント

各Wijmoコントロールバインディングは、 **initialized**イベントとブール型の**isInitialized**プロパティを公開しています。このイベントは、バインディングによってコントロールが作成され、バインディング属性で指定された値によってコントロールが初期化された直後に発生します。子バインディングを含むバインディング（たとえば、**wjFlexGrid**バインディングに子**wjFlexGridColumn**バインディングが含まれる場合）では、このイベントは子バインディングの初期化が完了し、親バインディングによって表されたコントロールに子バインディングが適用されたことを意味します。isInitializedプロパティは、initializedイベントが発生する直前にtrueに設定されます。ビューモデルのオブザーバブルプロパティをバインディングの **isInitialized**プロパティにバインドして、その値にアクセスできます。

以下の例では、バインディングによるコントロールの初期化が完了した後にFlexGridColumnの書式を調整しています。これにより、バインディングで定義された書式によってこれらの書式が上書きされないことが保証されます。

```
<!-- HTML -->
<div data-bind="'wjFlexGrid':{ itemsSource: dataArray, initialized: flexInitialized }">
     <div data-bind="wjFlexGridColumn:{ binding:'sales', format:'n2' }"></div>
     <div data-bind="wjFlexGridColumn:{ binding:'downloads', format:'n2' }"></div>
</div>


// ビューモデル
this.flexInitialized = function (data, sender, args) {
    var columns = sender.columns;
    for (var i = 0; i < columns.length; i++) {
        if (columns[i].dataType = wijmo.DataType.Number) {
            columns[i].format = 'n0?f;
        }
    }
}
```

カスタム要素のプリプロセッサ

Wijmo Knockoutプリプロセッサは、標準のKnockout **ko.bindingProvider.instance.preprocessNode** APIを使用します。Knockoutでは単一のインスタンスプロパティによってプリプロセッサ関数をアタッチするようになっており、新しいプリプロセッサを登録するとそれまでのプリプロセッサの登録が解除されます。そのため、同じページで他のカスタムプリプロセッサを使用した場合に問題が起こる可能性があります。

アタッチされている別のプリプロセッサに従うため、Wijmo Knockoutプリプロセッサは初期化中に現在登録されているプリプロセッサを保存し、別の処理ノードがWijmoコントロール要素として認識されていない場合はそのプリプロセッサに処理を委ねます。したがって、プリプロセッサのスタックが形成されます。ただし、Wijmo for Knockoutプリプロセッサの後（すなわち、<script>で**wijmo.knockout.js**モジュールへの参照を実行した後）に別のプリプロセッサを登録する場合は、開発者側でそのプリプロセッサも同様に動作するようにする必要があります。そうしなければ、Wijmo Knockoutプリプロセッサは無効になります。

Wijmo Knockoutプリプロセッサを意図的に無効にする場合は、コアWijmoモジュールへの参照を記述してから**wijmo.knockout.js**モジュールへの参照を記述するまでの間に、**wijmo.disableKnockoutTags**プロパティをtrueに設定します。以下に例を示します。

```
<script src="scripts/wijmo.js"></script>
<script src="scripts/wijmo.input.js"></script>

<script>
    wijmo.disableKnockoutTags = true;
</script>
<script src="scripts/wijmo.knockout.js"></script>
```

この場合、Wijmoコントロールをページマークアップに追加するときは常に従来のdata-bind構文を使用しなければならないことに注意してください。Wijmoカスタム要素は認識されません。

クラス

- wjAutoComplete
- wjBulletGraph

- wjCalendar
- wjCollectionViewNavigator
- wjCollectionViewPager
- wjColorPicker
- wjComboBox
- wjContextMenu
- wjFinancialChart
- wjFinancialChartSeries
- wjFlexChart
- wjFlexChartAnimation
- wjFlexChartAnnotation
- wjFlexChartAnnotationLayer
- wjFlexChartAtr
- wjFlexChartAxis
- wjFlexChartBollingerBands
- wjFlexChartCci
- wjFlexChartDataPoint
- wjFlexChartEnvelopes
- wjFlexChartFibonacci
- wjFlexChartFibonacciArcs
- wjFlexChartFibonacciFans
- wjFlexChartFibonacciTimeZones
- wjFlexChartGestures
- wjFlexChartLegend
- wjFlexChartLineMarker
- wjFlexChartMacd
- wjFlexChartMacdHistogram
- wjFlexChartMovingAverage
- wjFlexChartParametricFunctionSeries
- wjFlexChartPlotArea
- wjFlexChartRangeSelector
- wjFlexChartRsi
- wjFlexChartSeries
- wjFlexChartStochastic
- wjFlexChartTrendLine
- wjFlexChartWaterfall
- wjFlexChartWilliamsR
- wjFlexChartYFunctionSeries
- wjFlexGrid
- wjFlexGridColumn
- wjFlexGridFilter
- wjFlexPie
- wjFlexSheet
- wjGroupPanel
- wjInputColor
- wjInputDate
- wjInputDateTime
- wjInputMask
- wjInputNumber

- wjInputTime
- wjLinearGauge
- wjListBox
- wjMenu
- wjMenuItem
- wjMenuSeparator
- wjMultiAutoComplete
- wjMultiRow
- wjMultiSelect
- wjPivotChart
- wjPivotGrid
- wjPivotPanel
- wjPopup
- wjRadialGauge
- wjRange
- wjSheet
- wjSlicer
- wjStyle
- wjTooltip
- wjTreeView

wjAutoComplete クラス

ファイル	wijmo.knockout.js
モジュール	**wijmo.knockout**
基本クラス	**wjComboBox**
派生クラス	**wjMultiAutoComplete**
表示	☐継承されたメンバー ☐イベント発生元

AutoComplete コントロール用のKnockoutJSバインディング。

wjAutoComplete バインディングは、KnockoutJSアプリケーションに**AutoComplete** コントロールを追加する場合に使用します。次に例を示します。

```
<p>AutoCompleteコントロール:</p>
<div data-bind="wjAutoComplete:{
  itemsSource: countries,
  text: theCountry,
  isEditable: false,
  placeholder:'country' }">
</div>
```

wjAutoCompleteバインディングは、**AutoComplete** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**

wjBulletGraph クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
基本クラス **wjLinearGauge**
表示 ☐継承されたメンバー ☐イベント発生元

BulletGraph コントロール用のKnockoutJSバインディング。

wjBulletGraph バインディングは、KnockoutJSアプリケーションに**BulletGraph** コントロールを追加する場合に使用します。次に例を示します。

```
<p>BulletGraphコントロール:</p>
<div data-bind="wjBulletGraph:{
        value: props.value,
        min: props.min,
        max: props.max,
        format: props.format,
        good: props.ranges.middle.max,
        bad: props.ranges.middle.min,
        target: props.ranges.target,
        showRanges: props.showRanges }"
        class="linear-gauge">
    <div data-bind="wjRange:{
            wjProperty:'pointer',
            thickness: props.ranges.pointerThickness }">
    </div>
</div>
```

wjBulletGraphバインディングには、子バインディングとして **wjRange** を含めることができます。

wjBulletGraphバインディングは、**BulletGraph** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。 **value**プロパティは双方向にバインドされます。

wjCalendar クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

Calendar コントロール用のKnockoutJSバインディング。

wjCalendar バインディングは、KnockoutJSアプリケーションに**Calendar** コントロールを追加する場合に使用します。次に例を示します。

```
<p>Calendarコントロール:</p>
<div
  data-bind="wjCalendar:{ value: theDate }">
</div>
```

wjCalendarバインディングは、**Calendar** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **value**
- **displayMonth**

wjCollectionViewNavigator クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

ICollectionView のナビゲーター要素用のKnockoutJSバインディング。

wjCollectionViewNavigator ディレクティブを使用すると、**ICollectionView** の項目間を移動するための要素を追加できます。次に例を示します。

```
CollectionViewNavigator:</p>
<div
  data-bind="wjCollectionViewNavigator:{ cv: myCollectionView }">
</div>
```

wjCollectionViewNavigator ディレクティブには以下の1つの属性があります。

cv ナビゲートする**ICollectionView** オブジェクトへの参照。

wjCollectionViewPager クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

ICollectionView のページャー要素用のKnockoutJSバインディング。

wjCollectionViewPager ディレクティブを使用すると、ページ分割された**ICollectionView** のページ間を移動するための要素を追加できます。次に例を示します。

```
CollectionViewPager:</p>
<div
  data-bind="wjCollectionViewPager:{ cv: myCollectionView }">
</div>
```

wjCollectionViewPager ディレクティブには以下の1つの属性があります。

cv ナビゲートするページ分割された**ICollectionView** オブジェクトへの参照。

wjColorPicker クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

ColorPicker コントロール用のKnockoutJSバインディング。

wjColorPicker バインディングは、KnockoutJSアプリケーションに**ColorPicker** コントロールを追加する場合に使用します。次に例を示します。

```
<p>ColorPickerコントロール:</p>
<div
  data-bind="wjColorPicker:{ value: theColor }">
</div>
```

wjColorPickerバインディングは、**ColorPicker** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **value**

wjComboBox クラス

ファイル	wijmo.knockout.js
モジュール	**wijmo.knockout**
派生クラス	**wjAutoComplete**, **wjInputTime**, **wjMenu**, **wjMultiSelect**
表示	☐継承されたメンバー ☐イベント発生元

ComboBox コントロール用のKnockoutJSバインディング。

wjComboBox バインディングは、KnockoutJSアプリケーションに**ComboBox** コントロールを追加する場合に使用します。次に例を示します。

```
<p>ComboBoxコントロール:</p>
<div data-bind="wjComboBox:{
  itemsSource: countries,
  text: theCountry,
  isEditable: false,
  placeholder:'country' }">
</div>
```

wjComboBoxバインディングは、**ComboBox** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**

wjContextMenu クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

コンテキストメニューに使用されるKnockoutJS連結。

wjContextMenu 連結を使用して、ページ内の要素にコンテキストメニューを 追加できます。 **wjContextMenu** 連結は、**wjMenu** に基づいています。 ユーザーが要素に対してコンテキストメニューを要求したときに（通常は 右クリックしたとき）、ポップアップメニューを表示します。

wjContextMenu連結は、コンテキストメニューの適用先の 要素に追加されるパラメータとして指定されます。パラメータ値は、 メニューを含む要素のCSSセレクタです。次に例を示します。

```
<!-- コンテキストメニュー付きの段落 -->
<p data-bind="wjContextMenu:{ id:'#idMenu'}" >
 この段落にはコンテキストメニューがあります。</p>

<!-- コンテキストメニューを定義します（非表示、ID付き）-->
<div id="contextmenu" data-bind="wjMenu:{ header:'ファイル', itemClicked: menuItemClicked}">
    <span data-bind="wjMenuItem:{}">New</span>
    <span data-bind="wjMenuItem:{}">既存のファイルまたはフォルダを開く</span>
    <span data-bind="wjMenuItem:{}">現在のファイルを保存</span>
    <span data-bind="wjMenuSeparator:{}"></span>
    <span data-bind="wjMenuItem:{}">アプリケーションを終了</span>
</div>
```

wjFinancialChart クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

see:FinancialChart コントロールに使用されるKnockoutJS連結。

wjFinancialChart 連結を使用して、KnockoutJSアプリケーションに**FinancialChart** コントロールを追加できます。次に例を示します。

```
<p>FinancialChartコントロールの例：</p>
<div data-bind="wjFinancialChart:{ itemsSource: data, chartType:'Candlestick' }">
    <div data-bind="wjFlexChartLegend :{
        position:'Top' }">
    </div>
    <div data-bind="wjFinancialChartSeries:{
         name:'close',
        binding:'high,low,open,close' }">
    </div>
    </div>
</div>
```

wjFinancialChart連結には、**wjFlexChartAxis**、**wjFinancialChartSeries**、 **wjFlexChartLegend** の各子連結を含めることができます。

wjFinancialChart連結は、 **FinancialChart** コントロールのすべての読み取り/書き込み可能プロパティおよびイベント、さらに**FinancialChart.tooltip.content**プロパティに値を割り当てる**tooltipContent**プロパティをサポートします。 **selection**プロパティは、双方向連結モードを提供します。

wjFinancialChartSeries クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart FinancialSeries オブジェクトに対応するKnockoutJS連結。

WjFinancialChartSeries 連結は、**wjFinancialChart** 連結に含める必要があります。

WjFinancialChartSeries連結は、 **FinancialSeries** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。 **visibility**プロパティは、双方向連結モードを提供します。

wjFlexChart クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart コントロール用のKnockoutJSバインディング。

wjFlexChart バインディングは、KnockoutJSアプリケーションに**FlexChart** コントロールを追加する場合に使用します。次に例を示します。

```
<p>FlexChartコントロール:</p>
<div data-bind="wjFlexChart:{ itemsSource: data }">
    <div data-bind="wjFlexChartLegend :{
        position:'Top' }">
    </div>
    <div data-bind="wjFlexChartAxis:{
        wjProperty:'axisX',
        title: chartProps.titleX }">
    </div>
    <div data-bind="wjFlexChartAxis:{
        wjProperty:'axisY',
        majorUnit:5000 }">
    </div>
    <div data-bind="wjFlexChartSeries:{
        name:'Sales',
        binding:'sales' }">
    </div>
    <div data-bind="wjFlexChartSeries:{
        name:'Expenses',
        binding:'expenses' }">
    </div>
    <div data-bind="wjFlexChartSeries:{
        name:'Downloads',
        binding:'downloads',
        chartType:'LineSymbols' }">
    </div>
</div>
```

wjFlexChartバインディングには、子バインディングとして**wjFlexChartAxis**、**wjFlexChartSeries**、および**wjFlexChartLegend** を含めることができます。

wjFlexChartバインディングは、**FlexChart** コントロールのすべての読み取り/書き込みプロパティおよびイベントに加えて、 **FlexChart.tooltip.content**プロパティに値を割り当てる**tooltipContent**プロパティもサポートします。 **selection**プロパティは双方向にバインドされます。

wjFlexChartAnimation クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

ChartAnimation オブジェクトに使用されるKnockoutJS連結。

wjFlexChartAnimation 連結を使用して、KnockoutJSアプリケーションに**ChartAnimation** オブジェクトを追加できます。次に例を示します。

```
<p>ChartAnimationの例：</p>
<div data-bind="wjFlexChart:{ itemsSource: data, bindingX:'country',chartType:'Column' }">
    <div data-bind="wjFlexChartAxis:{ wjProperty:'axisX', title:'country' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Sales', binding:'sales' }"></div>
    <div data-bind="wjFlexChartAnimation:{ animationMode:'Series',easing:'Swing',duration:2000 }  "></div>
</div>
```

wjFlexChartAnimation連結は、 **ChartAnimation** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartAnnotation クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

注釈に使用されるKnockoutJS連結。

wjFlexChartAnnotationは、 **wjFlexChartAnnotationLayer** 連結に含める必要があります。次に例を示します。

```
<p>AnnotationLayerの例：</p>
<div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date', chartType:'Candlestick' }">
   <div data-bind="wjFinancialChartSeries:{ bindingX:'date', binding:'high,low,open,close' }"></div>
   <div data-bind="wjFlexChartAnnotationLayer:{}">
       <div data-bind="wjFlexChartAnnotation:{ type:'Rectangle', content:'E',height:20, width:20,attachment:'DataIndex',pointIndex:10}"></di
v>
       <div data-bind="wjFlexChartAnnotation:{ type:'Ellipse', content:'E',height:20, width:20,attachment:'DataIndex',pointIndex:30}"></div>
   </div>
      </div>
```

wjFlexChartAnnotationは、 **Circle**、**Rectangle**、**Polygon**など、使用可能な すべての注釈の形状タイプを表すために使用されます。注釈の形状は、 **type** 属性で指定されます。

wjFlexChartAnnotationLayer クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

AnnotationLayer オブジェクトに使用されるKnockoutJS連結。

wjFlexChartAnnotationLayer 連結を使用して、KnockoutJSアプリケーションに**AnnotationLayer** オブジェクトを追加できます。次に例を示します。

```
<p>AnnotationLayerの例：</p>
<div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date', chartType:'Candlestick' }">
   <div data-bind="wjFinancialChartSeries:{ bindingX:'date', binding:'high,low,open,close' }"></div>
   <div data-bind="wjFlexChartAnnotationLayer:{}">
       <div data-bind="wjFlexChartAnnotation:{ type:'Rectangle', content:'E',height:20, width:20,attachment:'DataIndex',pointIndex:10}"></di
v>
       <div data-bind="wjFlexChartAnnotation:{ type:'Ellipse', content:'E',height:20, width:20,attachment:'DataIndex',pointIndex:30}"></div>
   </div>
      </div>
```

wjFlexChartAnnotationLayer連結は、 **AnnotationLayer** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartAtr クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

ATR オブジェクトに使用されるKnockoutJS連結。

wjFlexChartAtr 連結を使用して、KnockoutJSアプリケーションに**ATR** オブジェクトを追加できます。次に例を示します。

```
<p>ATRの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date'}">
        <div data-bind="wjFlexChartAtr:{ binding:'high,low,open,close',period:'14' }"></div>
  </div>
```

wjFlexChartAtr連結は、 **ATR** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartAxis クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart の**Axis** オブジェクト用のKnockoutJSバインディング。

wjFlexChartAxis バインディングは**wjFlexChart** バインディング内に記述する必要があります。**wjProperty**属性を使用して、このバインディングで初期化するプロパティ（**axisX**または**axisY**）を指定します。

wjFlexChartAxisバインディングは、**Axis** クラスのすべての読み取り/書き込みプロパティおよびイベントをサポートします。

wjFlexChartBollingerBands クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

BollingerBands オブジェクトに使用されるKnockoutJS連結。

wjFlexChartBollingerBands 連結を使用して、KnockoutJSアプリケーションに**BollingerBands** オブジェクトを追加できます。次に例を示します。

```
<p>BollingerBandsの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date'}">
        <div data-bind="wjFlexChartStochastic:{ binding:'high,low,open,close',kPeriod:14,dPeriod:3,smoothingPeriod:1 }" ></div>
  </div>
```

wjFlexChartBollingerBands連結は、 **BollingerBands** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartCci クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

CCI オブジェクトに使用されるKnockoutJS連結。

wjFlexChartCci 連結を使用して、KnockoutJSアプリケーションに**CCI** オブジェクトを追加できます。次に例を示します。

```
<p>CCIの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date'}">
        <div data-bind="wjFlexChartCci:{ binding:'high,low,open,close',period:20 }"></div>
  </div>
```

wjFlexChartCci連結は、 **CCI** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartDataPoint クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

DataPoint オブジェクトに使用されるKnockoutJS連結。 **wjFlexChartDataPoint**は、 **wjFlexChartAnnotation** に含める必要があります。**wjFlexChartDataPoint**によって値が割り当てられる親オブジェクトのプロパティは、 **wjProperty**属性で指定されます。

wjFlexChartDataPoint 連結を使用して、KnockoutJSアプリケーションに**DataPoint** オブジェクトを追加できます。次に例を示します。

```
<p>DataPointの例：</p>
  <div data-bind="wjFlexChartDataPoint{ wjProperty:'point', x:0.9, y:0.4}" ></div>
```

wjFlexChartDataPoint連結は、 **DataPoint** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartEnvelopes クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

Envelopes オブジェクトに使用されるKnockoutJS連結。

wjFlexChartEnvelopes 連結を使用して、KnockoutJSアプリケーションに**Envelopes** オブジェクトを追加できます。次に例を示します。

```
<p>Envelopesの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date'}">
        <div data-bind="wjFlexChartEnvelopes:{ binding:'close', type:'Simple', size:0.03, period:20}" ></div>
  </div>
```

wjFlexChartEnvelopes連結は、 **Envelopes** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartFibonacci クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

Fibonacci オブジェクトに使用されるKnockoutJS連結。

wjFlexChartFibonacci 連結を使用して、KnockoutJSアプリケーションに**Fibonacci** オブジェクトを追加できます。次に例を示します。

```
<p>Fibonacciの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date', chartType:'Candlestick' }">
        <div data-bind="wjFinancialChartSeries:{ bindingX:'date', binding:'high,low,open,close' }"></div>
        <div data-bind="wjFlexChartFibonacci:{ binding:'close', symbolSize:1, labelPosition:'Left',  uptrend: true}"></div>
  </div>
```

wjFlexChartFibonacci連結は、 **Fibonacci** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartFibonacciArcs クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FibonacciArcs オブジェクトに使用されるKnockoutJS連結。

wjFlexChartFibonacciArcs 連結を使用して、KnockoutJSアプリケーションに**FibonacciArcs** オブジェクトを追加できます。次に例を示します。

```
<p>FibonacciArcsの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date', chartType:'Candlestick' }">
        <div data-bind="wjFinancialChartSeries:{ bindingX:'date', binding:'high,low,open,close' }"></div>
        <div data-bind="wjFlexChartFibonacciArcs:{ binding:'close', start:start, end: end,  labelPosition:'Top'}"></div>
  </div>
```

wjFlexChartFibonacciArcs連結は、 **FibonacciArcs** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartFibonacciFans クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FibonacciFans オブジェクトに使用されるKnockoutJS連結。

wjFlexChartFibonacciFans 連結を使用して、KnockoutJSアプリケーションに**FibonacciFans** オブジェクトを追加できます。次に例を示します。

```
<p>FibonacciFansの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date', chartType:'Candlestick' }">
        <div data-bind="wjFinancialChartSeries:{ bindingX:'date', binding:'high,low,open,close' }"></div>
        <div data-bind="wjFlexChartFibonacciFans:{ binding:'close', start:start, end: end,  labelPosition:'Top'}"></div>
  </div>
```

wjFlexChartFibonacciFans連結は、 **FibonacciFans** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartFibonacciTimeZones クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FibonacciTimeZones オブジェクトに使用されるKnockoutJS連結。

wjFlexChartFibonacciTimeZones 連結を使用して、KnockoutJSアプリケーションに**FibonacciTimeZones** オブジェクトを追加できます。次に例を示します。

```
<p>FibonacciTimeZonesの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date', chartType:'Candlestick' }">
        <div data-bind="wjFinancialChartSeries:{ bindingX:'date', binding:'high,low,open,close' }"></div>
        <div data-bind="wjFlexChartFibonacciTimeZones:{ binding:'close', startX:zStart, endX: zEnd,  labelPosition:'Right'}"></div>
  </div>
```

wjFlexChartFibonacciTimeZones連結は、 **FibonacciTimeZones** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartGestures クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

ChartGestures オブジェクトに使用されるKnockoutJS連結。

wjFlexChartGestures 連結を使用して、KnockoutJSアプリケーションに**ChartGestures** コントロールを追加できます。次に例を示します。

```
<p>ChartGesturesの例：</p>
<div data-bind="wjFlexChart:{ itemsSource: data, bindingX:'country' }">
    <div data-bind="wjFlexChartAxis:{ wjProperty:'axisX', title:'country' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Sales', binding:'sales' }"></div>
    <div data-bind="wjFlexChartGestures:{ scaleX:0.5, posX:0.1 } "></div>
</div>
```

wjFlexChartGestures連結は、 **ChartGestures** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartLegend クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

チャートの**Legend** オブジェクト用のKnockoutJSバインディング。

wjFlexChartLegend バインディングは**wjFlexChart** または**wjFlexPie** のいずれかのバインディング内に記述する必要があります。

wjFlexChartLegendバインディングは、**Legend** クラスのすべての読み取り/書き込みプロパティおよびイベントをサポートします。

wjFlexChartLineMarker クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

LineMarker コントロールに使用されるKnockoutJS連結。

wjFlexChartLineMarker 連結を使用して、KnockoutJSアプリケーションに**LineMarker** コントロールを追加できます。次に例を示します。

```
<p>LineMarkerの例：</p>
<div data-bind="wjFlexChart:{ itemsSource: data, bindingX:'country' }">
    <div data-bind="wjFlexChartAxis:{ wjProperty:'axisX', title:'country' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Sales', binding:'sales' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Expenses', binding:'expenses' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Downloads', binding:'downloads' }"></div>
    <div data-bind="wjFlexChartLineMarker:{ interaction:'Move', lines:'Both' }"></div>
</div>
```

wjFlexChartLineMarker連結は、 **LineMarker** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartMacd クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

Macd オブジェクトに使用されるKnockoutJS連結。

wjFlexChartMacd 連結を使用して、KnockoutJSアプリケーションに**Macd** オブジェクトを追加できます。次に例を示します。

```
<p>Macdの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date'}">
        <div data-bind="wjFlexChartMacd:{ binding:'close',fastPeriod:12, slowPeriod:26,smoothingPeriod:9 }" ></div>
  </div>
```

wjFlexChartMacd連結は、 **Macd** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartMacdHistogram クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

MacdHistogram オブジェクトに使用されるKnockoutJS連結。

wjFlexChartMacdHistogram 連結を使用して、KnockoutJSアプリケーションに**MacdHistogram** オブジェクトを追加できます。次に例を示します。

```
<p>MacdHistogramの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date'}">
        <div data-bind="WjFlexChartMacdHistogram:{ binding:'close',fastPeriod:12, slowPeriod:26,smoothingPeriod:9 }" ></div>
  </div>
```

wjFlexChartMacdHistogram連結は、 **MacdHistogram** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartMovingAverage クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

MovingAverage オブジェクトに使用されるKnockoutJS連結。

wjFlexChartMovingAverage 連結を使用して、KnockoutJSアプリケーションに**MovingAverage** オブジェクトを追加できます。次に例を示します。

```
<p>MovingAverageの例：</p>
<div data-bind="wjFlexChart:{ itemsSource: trendItemsSource, bindingX:'x' }">
    <div data-bind="wjFlexChartAxis:{ wjProperty:'axisX', title:'country' }"></div>
    <div data-bind="wjFlexChartSeries:{ chartType:'Scatter', name:'Base Data', binding:'y' }"></div>
    <div data-bind="wjFlexChartMovingAverage:{ binding:'y', bindingX:'x', period:2 }  "></div>
</div>
```

wjFlexChartMovingAverage連結は、 **MovingAverage** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartParametricFunctionSeries クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

ParametricFunctionSeries オブジェクトに使用されるKnockoutJS連結。

wjFlexChartParametricFunctionSeries 連結を使用して、KnockoutJSアプリケーションに**ParametricFunctionSeries** オブジェクトを追加できます。次に例を示します。

```
<p>ParametricFunctionSeriesの例：</p>
<div data-bind="wjFlexChart:{ itemsSource: trendItemsSource, bindingX:'x' }">
    <div data-bind="wjFlexChartSeries:{ name:'Sales', binding:'sales' }"></div>
    <div data-bind="wjFlexChartParametricFunctionSeries:{  sampleCount:1000, max: max,xFunc:xFunc,yFunc:yFunc  }"></div>
</div>
```

wjFlexChartParametricFunctionSeries連結は、 **ParametricFunctionSeries** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartPlotArea クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

PlotArea オブジェクトに使用されるKnockoutJS連結。

wjFlexChartPlotArea 連結を使用して、KnockoutJSアプリケーションに**PlotArea** オブジェクトを追加できます。次に例を示します。

```
<p>PlotAreaの例：</p>
<div data-bind="wjFlexChart:{ itemsSource: data, bindingX:'country' }">
    <div data-bind="wjFlexChartAxis:{ wjProperty:'axisX', title:'country' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Sales', binding:'sales' }"></div>
    <div data-bind="wjFlexChartPlotArea:{ row:0, name:'plot1', style:{ fill:'rgba(136,189,230,0.2)'} }  "></div>
</div>
```

wjFlexChartPlotArea連結は、 **PlotArea** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartRangeSelector クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

RangeSelector コントロールに使用されるKnockoutJS連結。

wjFlexChartRangeSelector 連結を使用して、KnockoutJSアプリケーションに**RangeSelector** コントロールを追加できます。次に例を示します。

```
<p>RangeSelectorコントロールの例：</p>
<div data-bind="wjFlexChart:{ itemsSource: data, bindingX:'country' }">
    <div data-bind="wjFlexChartAxis:{ wjProperty:'axisX', title:'country' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Sales', binding:'sales' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Expenses', binding:'expenses' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Downloads', binding:'downloads' }"></div>
    <div data-bind="wjFlexChartRangeSelector:{ seamless:'true',rangeChanged: rangeChanged }"></div>
</div>
```

wjFlexChartRangeSelector連結は、 **RangeSelector** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartRsi クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

RSI オブジェクトに使用されるKnockoutJS連結。

wjFlexChartRsi 連結を使用して、KnockoutJSアプリケーションに**RSI** オブジェクトを追加できます。次に例を示します。

```
<p>RSIの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date', chartType:'Candlestick' }">
        <div data-bind="wjFlexChartRsi:{ binding:'high,low,open,close',period:20 }"></div>
  </div>
```

wjFlexChartRsi連結は、 **RSI** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartSeries クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart の**Series** オブジェクト用のKnockoutJSバインディング。

wjFlexChartSeries バインディングは**wjFlexChart** バインディング内に記述する必要があります。

wjFlexChartSeriesバインディングは、**Series** クラスのすべての読み取り/書き込みプロパティおよびイベントをサポートします。 **visibility**プロパティは双方向にバインドされます。

wjFlexChartStochastic クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

Stochastic オブジェクトに使用されるKnockoutJS連結。

wjFlexChartStochastic 連結を使用して、KnockoutJSアプリケーションに**Stochastic** オブジェクトを追加できます。次に例を示します。

```
<p>Stochasticの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date'}">
        <div data-bind="wjFlexChartStochastic:{ binding:'high,low,open,close',kPeriod:14,dPeriod:3,smoothingPeriod:1 }" ></div>
  </div>
```

wjFlexChartStochastic連結は、 **Stochastic** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartTrendLine クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

TrendLine オブジェクトに使用されるKnockoutJS連結。

wjFlexChartTrendLine 連結を使用して、KnockoutJSアプリケーションに**TrendLine** オブジェクトを追加できます。次に例を示します。

```
<p>TrendLineの例：</p>
<div data-bind="wjFlexChart:次に例を示します。{ itemsSource: data, bindingX:'country',chartType:'Column' }">
    <div data-bind="wjFlexChartAxis:{ wjProperty:'axisX', title:'country' }"></div>
    <div data-bind="wjFlexChartSeries:{ name:'Sales', binding:'sales' }"></div>
    <div data-bind="wjFlexChartAnimation:{ animationMode:'Series',easing:'Swing',duration:2000 }  "></div>
</div>
```

wjFlexChartTrendLine連結は、 **TrendLine** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartWaterfall クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

Waterfall オブジェクトに使用されるKnockoutJS連結。

wjFlexChartWaterfall 連結を使用して、KnockoutJSアプリケーションに**Waterfall** オブジェクトを追加できます。次に例を示します。

```
<p>Waterfallの例：</p>
<div data-bind="wjFlexChart:{ itemsSource: trendItemsSource,  binding:'value',bindingX:'name' }">
    <div data-bind="wjFlexChartWaterfall:{  relativeData:true, connectorLines: true, start:1000,showIntermediateTotal: true,
                    intermediateTotalPositions:[3, 6, 9, 12], intermediateTotalLabels:['Q1', 'Q2', 'Q3', 'Q4'],name:'Increase,Decrease,Tot
al'}"></div>
</div>
```

wjFlexChartWaterfall連結は、 **Waterfall** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartWilliamsR クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

WilliamsR オブジェクトに使用されるKnockoutJS連結。

wjFlexChartWilliamsR 連結を使用して、KnockoutJSアプリケーションに**WilliamsR** オブジェクトを追加できます。次に例を示します。

```
<p>WilliamsRの例：</p>
   <div data-bind="wjFinancialChart:{ itemsSource: fData, bindingX:'date'}">
        <div data-bind="wjFlexChartWilliamsR:{ binding:'high,low,open,close',period:20 }"></div>
  </div>
```

wjFlexChartWilliamsR連結は、 **WilliamsR** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexChartYFunctionSeries クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

YFunctionSeries オブジェクトに使用されるKnockoutJS連結。

wjFlexChartYFunctionSeries 連結を使用して、KnockoutJSアプリケーションに**YFunctionSeries** オブジェクトを追加できます。次に例を示します。

```
<p>YFunctionSeriesの例：</p>
<div data-bind="wjFlexChart:{ itemsSource: trendItemsSource, bindingX:'x' }">
    <div data-bind="wjFlexChartYFunctionSeries:{  min:10, max:-10, sampleCount:100,func:func }"></div>
</div>
```

wjFlexChartYFunctionSeries連結は、 **YFunctionSeries** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjFlexGrid クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
派生クラス **wjFlexSheet**, **wjMultiRow**, **wjPivotGrid**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid コントロール用のKnockoutJSバインディング。

wjFlexGrid バインディングは、KnockoutJSアプリケーションに**FlexGrid** コントロールを追加する場合に使用します。例:

```
<p>FlexGridコントロール:</p>
<div data-bind="wjFlexGrid:{ itemsSource: data }">
    <div data-bind="wjFlexGridColumn:{
        header:'Country',
        binding:'country',
        width:'*' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Date',
        binding:'date' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Revenue',
        binding:'amount',
        format:'n0' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Active',
        binding:'active' }">
    </div>
</div>
```

wjFlexGridバインディングには、子バインディングとして **wjFlexGridColumn** を含めることができます。

wjFlexGridバインディングは、**scrollPosition**プロパティ、**selection**プロパティおよび**columnLayout**プロパティを除く**FlexGrid** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。

wjFlexGridColumn クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid の**Column** オブジェクト用のKnockoutJSバインディング。

wjFlexGridColumn バインディングは**wjFlexGrid** バインディング内に記述する必要があります。次に例を示します。

```
<p>FlexGridコントロール:</p>
<div data-bind="wjFlexGrid:{ itemsSource: data }">
    <div data-bind="wjFlexGridColumn:{
        header:'Country',
        binding:'country',
        width:'*' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Date',
        binding:'date' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Revenue',
        binding:'amount',
        format:'n0' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Active',
        binding:'active' }">
    </div>
</div>
```

wjFlexGridColumnバインディングは、**Column** クラスのすべての読み取り/書き込みプロパティおよびイベントをサポートします。 **isSelected**プロパティは双方向にバインドされます。

wjFlexGridColumn バインディングには、**Column**クラスのプロパティと一致する通常の属性に加えて、条件付き書式を提供する **wjStyle** バインディングと、セルテンプレートとして使用されるHTMLフラグメントを含めることができます。グリッドの行は、カスタムセルの内容を収めるために自動的に高さが拡張されます。

wjStyleバインディングおよびHTMLフラグメントでは、現在の行にバインドされた項目を表すKnockoutJSバインディングの **$item**オブザーバブル変数を使用できます。また、セルの行インデックスと列インデックスを含む**$row**および**$col**オブザーバブル変数も使用できます。次に例を示します。

```
<div data-bind="wjFlexGridColumn:{
        header:'Symbol',
        binding:'symbol',
        readOnly: true,
        width:'*' }">
  <a data-bind="attr:{
        href:'https://finance.yahoo.com/q?s=' + $item().symbol() },
        text:$item().symbol">
  </a>
</div>
<div data-bind="wjFlexGridColumn:{
     header:'Change',
        binding:'changePercent',
        format:'p2',
        width:'*'
        },
        wjStyle:{
        color: getAmountColor($item().change) }">
</div>
```

これらのバインディングは2つの列を作成します。 最初の列は、バインドされた項目の"symbol"プロパティに基づいてハイパーリンクを生成するテンプレートを持ちます。 2番目の列は、コントローラーに実装された関数によって決定された色で値をレンダリングする条件付きスタイルを持ちます。

wjFlexGridFilter クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid の**FlexGridFilter** オブジェクト用のKnockoutJSバインディング。

wjFlexGridFilter バインディングは**wjFlexGrid** バインディング内に記述する必要があります。次に例を示します。

```
<p>列フィルタが設定されたFlexGridコントロール:</p>
<div data-bind="wjFlexGrid:{ itemsSource: data }">
    <div data-bind="wjFlexGridColumn:{ filterColumns:['country', 'amount']  }"></div>

    <div data-bind="wjFlexGridColumn:{
        header:'Country',
        binding:'country',
        width:'*' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Date',
        binding:'date' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Revenue',
        binding:'amount',
        format:'n0' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Active',
        binding:'active' }">
    </div>
</div>
```

wjFlexGridFilterバインディングは、**FlexGridFilter** クラスのすべての読み取り/書き込みプロパティおよびイベントをサポートします。

wjFlexPie クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FlexPie コントロール用のKnockoutJSバインディング。

wjFlexPie バインディングは、KnockoutJSアプリケーションに**FlexPie** コントロールを追加する場合に使用します。次に例を示します。

```
<p>FlexPieコントロール:</p>
<div data-bind="wjFlexPie:{
        itemsSource: data,
        binding:'value',
        bindingName:'name',
        header:'Fruit By Value' }">
    <div data-bind="wjFlexChartLegend :{ position:'Top' }"></div>
</div>
```

wjFlexPieバインディングには、子バインディングとして **wjFlexChartLegend** を含めることができます。

wjFlexPieバインディングは、**FlexPie** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。

wjFlexSheet クラス

ファイル	`wijmo.knockout.js`
モジュール	**wijmo.knockout**
基本クラス	**wjFlexGrid**
表示	☐継承されたメンバー ☐イベント発生元

FlexSheet コントロールのKnockoutJS連結。

wjFlexSheet 連結を使用して、KnockoutJSアプリケーションに**FlexSheet** コントロールを追加できます。

wjFlexSheet連結には、**wjSheet** 子連結を含めることができます。

wjFlexSheet連結は、**FlexSheet** コントロールのすべての読み取り/書き込み可能なプロパティおよびイベントをサポートします。

wjGroupPanel クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid の**GroupPanel** コントロール用のKnockoutJSバインディング。

wjGroupPanelバインディングは、**grid**プロパティを使用して**FlexGrid**コントロールに接続します。 次に例を示します。

```
<p>GroupPanelを含むFlexGridコントロール:</p>

<div data-bind="wjGroupPanel:{ grid: flex(), placeholder:'Drag columns here to create groups.'}"></div>

<div data-bind="wjFlexGrid:{ control: flex, itemsSource: data }">
    <div data-bind="wjFlexGridColumn:{
        header:'Country',
        binding:'country',
        width:'*' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Date',
        binding:'date' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Revenue',
        binding:'amount',
        format:'n0' }">
    </div>
    <div data-bind="wjFlexGridColumn:{
        header:'Active',
        binding:'active' }">
    </div>
</div>
```

wjGroupPanelバインディングは、**GroupPanel** クラスのすべての読み取り/書き込みプロパティおよびイベントをサポートします。

wjInputColor クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

InputColor コントロール用のKnockoutJSバインディング。

wjInputColor バインディングは、KnockoutJSアプリケーションに**InputColor** コントロールを追加する場合に使用します。次に例を示します。

```
<p>InputColorコントロール:</p>
<div
  data-bind="wjInputColor:{ value: theColor }">
</div>
```

wjInputColorバインディングは、**InputColor** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **isDroppedDown**
- **text**
- **value**

wjInputDate クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

InputDate コントロール用のKnockoutJSバインディング。

wjInputDate バインディングは、KnockoutJSアプリケーションに**InputDate** コントロールを追加する場合に使用します。次に例を示します。

```
<p>InputDateコントロール:</p>
<div data-bind="wjInputDate:{
  value: theDate,
  format:'M/d/yyyy' }">
</div>
```

wjInputDateバインディングは、**InputDate** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **isDroppedDown**
- **text**
- **value**

wjInputDateTime クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

InputDateTime コントロールのKnockoutJS連結。

wjInputDateTime 連結を使用して、KnockoutJSアプリケーションに**InputDateTime** コントロールを追加できます。

wjInputDateTime連結は、**InputDateTime** コントロールのすべての読み取り/書き込み可能なプロパティおよびイベントをサポートします。

wjInputMask クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

InputMask コントロール用のKnockoutJSバインディング。

wjInputMask バインディングは、KnockoutJSアプリケーションに**InputMask** コントロールを追加する場合に使用します。次に例を示します。

```
<p>InputMaskコントロール:</p>
<div data-bind="wjInputMask:{
  mask:'99/99/99',
  promptChar:'*' }">
</div>
```

wjInputMaskバインディングは、**InputMask** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。 **value**プロパティは双方向にバインドされます。

wjInputNumber クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

InputNumber コントロール用のKnockoutJSバインディング。

wjInputNumber バインディングは、KnockoutJSアプリケーションに**InputNumber** コントロールを追加する場合に使用します。次に例を示します。

```
<p>InputNumberコントロール:</p>
<div data-bind="wjInputNumber:{
  value: theNumber,
  min:0,
  max:10,
  format:'n0',
  placeholder:'number between zero and ten' }">
</div>
```

wjInputNumberバインディングは、**InputNumber** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **value**
- **text**

wjInputTime クラス

ファイル	wijmo.knockout.js
モジュール	**wijmo.knockout**
基本クラス	**wjComboBox**
表示	☐継承されたメンバー ☐イベント発生元

InputTime コントロール用のKnockoutJSバインディング。

wjInputTime バインディングは、KnockoutJSアプリケーションに**InputTime** コントロールを追加する場合に使用します。次に例を示します。

```
<p>InputTimeコントロール:</p>
<div data-bind="wjInputTime:{
  min: new Date(2014, 8, 1, 9, 0),
  max: new Date(2014, 8, 1, 17, 0),
  step:15,
  format:'h:mm tt',
  value: theDate }">
</div>
```

wjInputTimeバインディングは、**InputTime** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**
- **value**

wjLinearGauge クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
派生クラス **wjBulletGraph**
表示 ☐継承されたメンバー ☐イベント発生元

LinearGauge コントロール用のKnockoutJSバインディング。

wjLinearGauge バインディングは、KnockoutJSアプリケーションに**LinearGauge** コントロールを追加する場合に使用します。次に例を示します。

```
<p>LinearGaugeコントロール:</p>
<div data-bind="wjLinearGauge:{
        value: props.value,
        min: props.min,
        max: props.max,
        format: props.format,
        showRanges: props.showRanges }"
        <class="linear-gauge">
    <div data-bind="wjRange:{
            wjProperty:'pointer',
            thickness: props.ranges.pointerThickness }">
    </div>
    <div data-bind="wjRange:{
            min: props.ranges.lower.min,
            max: props.ranges.lower.max,
            color: props.ranges.lower.color }">
    </div>
    <div data-bind="wjRange:{
            min: props.ranges.middle.min,
            max: props.ranges.middle.max,
            color: props.ranges.middle.color }">
    </div>
    <div data-bind="wjRange:{
            min: props.ranges.upper.min,
            max: props.ranges.upper.max,
            color: props.ranges.upper.color }">
    </div>
</div>
```

wjLinearGaugeバインディングには、子バインディングとして **wjRange** を含めることができます。

wjLinearGaugeバインディングは、**LinearGauge** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。 **value**プロパティは双方向にバインドされます。

wjListBox クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

ListBox コントロール用のKnockoutJSバインディング。

wjListBox バインディングは、KnockoutJSアプリケーションに**ListBox** コントロールを追加する場合に使用します。次に例を示します。

```
<p>ListBoxコントロール:</p>
<div data-bind="wjListBox:{
  itemsSource: countries,
  selectedItem: theCountry }">
</div>
```

wjListBoxバインディングは、**ListBox** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **selectedIndex**
- **selectedItem**
- **selectedValue**

wjMenu クラス

ファイル	wijmo.knockout.js
モジュール	**wijmo.knockout**
基本クラス	**wjComboBox**
表示	☐継承されたメンバー ☐イベント発生元

Menu コントロール用のKnockoutJSバインディング。

wjMenu バインディングは、KnockoutJSアプリケーションに**Menu** コントロールを追加する場合に使用します。次に例を示します。

```
<p>Menuコントロール:</p>
<div data-bind="wjMenu:{ value: tax, header:'Tax' }">
    <span data-bind="wjMenuItem:{ value:0 }">免除</span>
    <span data-bind="wjMenuSeparator:{}"></span>
    <span data-bind="wjMenuItem:{ value:.05 }">5%</span>
    <span data-bind="wjMenuItem:{ value:.1 }">10%</span>
    <span data-bind="wjMenuItem:{ value:.15 }">15%</span>
</div>
```

wjMenuバインディングには、子バインディングとして**wjMenuItem** および**wjMenuSeparator** を含めることができます。

バインディングは、**Menu** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**
- **value**

wjMenuItem クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

メニュー項目用のKnockoutJSバインディング。

wjMenuItem バインディングは、**Menu** コントロールにメニュー項目を追加する場合に使用します。 **wjMenuItem** バインディングは**wjMenu** バインディング内に記述する必要があります。 次に例を示します。

```
<p>4つのメニュー項目を持つMenuコントロール:</p>
<div data-bind="wjMenu:{ value: tax, header:'Tax' }">
    <span data-bind="wjMenuItem:{ value:0 }">免除</span>
    <span data-bind="wjMenuItem:{ value:.05 }">5%</span>
    <span data-bind="wjMenuItem:{ value:.1 }">10%</span>
    <span data-bind="wjMenuItem:{ value:.15 }">15%</span>
</div>
```

wjMenuItemバインディングは以下の属性をサポートしています。

cmd 項目がクリックされたときにコントローラーで実行する関数。
cmdParam 項目がクリックされたときに**cmd**関数に渡されるパラメーター。
value 項目がクリックされたときに選択される値（このプロパティと**cmd**のどちらか一方を使用します）。

wjMenuSeparator クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

メニューセパレーター用のKnockoutJSバインディング。

wjMenuSeparator は選択できないセパレーターを**Menu** コントロールに追加するもので、属性はありません。 これは**wjMenu** バインディング内に記述する必要があります。次に例を示します。

```
<p>4つのメニュー項目と1つのセパレーターを持つMenuコントロール:</p>
<div data-bind="wjMenu:{ value: tax, header:'Tax' }">
    <span data-bind="wjMenuItem:{ value:0 }">免除</span>
    <span data-bind="wjMenuSeparator:{}"></span>
    <span data-bind="wjMenuItem:{ value:.05 }">5%</span>
    <span data-bind="wjMenuItem:{ value:.1 }">10%</span>
    <span data-bind="wjMenuItem:{ value:.15 }">15%</span>
</div>
```

wjMultiAutoComplete クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
基本クラス **wjAutoComplete**
表示 ☐継承されたメンバー ☐イベント発生元

MultiAutoComplete コントロールに使用されるKnockoutJS連結。

wjMultiAutoComplete 連結を使用して、KnockoutJSアプリケーションに**MultiAutoComplete** コントロールを追加できます。次に例を示します。

```
<p>MultiAutoCompleteコントロールの例：</p>
<div data-bind="MultiAutoComplete:{
  itemsSource: countries,
  maxSelectedItems:4,}">
</div>
```

wjMultiAutoComplete連結は、 **MultiAutoComplete** コントロールのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjMultiRow クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
基本クラス **wjFlexGrid**
表示 ☐継承されたメンバー ☐イベント発生元

MultiRow オブジェクトに使用されるKnockoutJS連結。 **wjMultiRow** 連結を使用して、KnockoutJSアプリケーションに**MultiRow** コントロールを追加できます。次に例を示します。   **wjMultiRow**連結は、 **MultiRow** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjMultiSelect クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
基本クラス **wjComboBox**
表示 ☐継承されたメンバー ☐イベント発生元

MultiSelect コントロールに使用されるKnockoutJS連結。

wjMultiSelect 連結を使用して、KnockoutJSアプリケーションに**MultiSelect** コントロールを追加できます。次に例を示します。

```
<p>MultiSelectコントロールの例：</p>
<div data-bind="MultiSelect:{
  itemsSource: countries,
  isEditable: false,
  headerFormat:'{count} countries selected' }">
</div>
```

wjMultiSelect連結は、 **MultiSelect** コントロールのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。次のプロパティは、双方向連結モードを提供します。

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**

wjPivotChart クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

PivotChart オブジェクトに使用されるKnockoutJS連結。 **wjPivotChart** 連結を使用して、KnockoutJSアプリケーションに**PivotChart** コントロールを追加できます。次に例を示します。   **wjPivotChart**連結は、 **PivotChart** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjPivotGrid クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
基本クラス **wjFlexGrid**
表示 ☐継承されたメンバー ☐イベント発生元

PivotGrid オブジェクトに使用されるKnockoutJS連結。 **wjPivotGrid** 連結を使用して、KnockoutJSアプリケーションに**PivotGrid** コントロールを追加できます。次に例を示します。   **wjPivotGrid**連結は、 **PivotGrid** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjPivotPanel クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

PivotPanel オブジェクトに使用されるKnockoutJS連結。 **wjPivotPanel** 連結を使用して、KnockoutJSアプリケーションに**PivotPanel** コントロールを追加できます。次に例を示します。   **wjPivotPanel**連結は、**PivotPanel** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjPopup クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

Popup コントロールに使用されるKnockoutJS連結。

wjPopup 連結を使用して、KnockoutJSアプリケーションに**Popup** コントロールを追加できます。次に例を示します。

```
<p>ボタンによってトリガされるPopupコントロールの例:</p>
<button id="btn2" type="button">
    クリックしてポップアップを表示
</button>
 <div class="popover" data-bind="wjPopup:{
      control: popup,
      owner:'#btn2',
      showTrigger:'Click',
      hideTrigger:'Click'}"
 >
<h3>
         ご挨拶
</h3>
 <div class="popover-content">
           {{firstName}} {{lastName}}さん、こんにちは
 </div>
</div>
```

wjRadialGauge クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

RadialGauge コントロール用のKnockoutJSバインディング。

wjRadialGauge バインディングは、KnockoutJSアプリケーションに**RadialGauge** コントロールを追加する場合に使用します。次に例を示します。

```
<p>RadialGaugeコントロール:</p>
<div data-bind="wjRadialGauge:{
        value: props.value,
        min: props.min,
        max: props.max,
        format: props.format,
        showRanges: props.showRanges }"
        class="radial-gauge">
    <div data-bind="wjRange:{
            wjProperty:'pointer',
            thickness: props.ranges.pointerThickness }">
    </div>
    <div data-bind="wjRange:{
            min: props.ranges.lower.min,
            max: props.ranges.lower.max,
            color: props.ranges.lower.color }">
    </div>
    <div data-bind="wjRange:{
            min: props.ranges.middle.min,
            max: props.ranges.middle.max,
            color: props.ranges.middle.color }">
    </div>
    <div data-bind="wjRange:{
            min: props.ranges.upper.min,
            max: props.ranges.upper.max,
            color: props.ranges.upper.color }">
    </div>
</div>
```

wjRadialGaugeバインディングには、子バインディングとして **wjRange** を含めることができます。

wjRadialGaugeバインディングは、**RadialGauge** コントロールのすべての読み取り/書き込みプロパティおよびイベントをサポートします。 **value**プロパティは双方向にバインドされます。

wjRange クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

ゲージの**Range** オブジェクト用のKnockoutJSバインディング。

wjRange バインディングは以下のいずれかのバインディング内に記述する必要があります。

- **wjLinearGauge**
- **wjRadialGauge**
- **wjBulletGraph**

デフォルトでは、このバインディングはChartコントロールの**ranges**コレクションに**Range**オブジェクトを追加します。**wjProperty**属性を使用して、このバインディングで初期化する別のChartプロパティ（たとえば、**pointer**プロパティなど）を指定できます。

wjRangeバインディングは、**Range** クラスのすべての読み取り/書き込みプロパティおよびイベントをサポートします。

wjSheet クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

FlexSheet Sheet オブジェクトに対応するKnockoutJS連結。

wjSheet 連結は、**wjFlexSheet** 連結に含める必要があります。

wjSheet連結は、**Sheet** クラスのすべての読み取り/書き込み可能なプロパティおよびイベントをサポートします。

wjSlicer クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

Slicer オブジェクトに使用されるKnockoutJS連結。 **wjSlicer** 連結を使用して、KnockoutJSアプリケーションに**Slicer** コントロールを追加できます。次に例を示します。   **wjSlicer** 連結は、 **Slicer** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wjStyle クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**

FlexGrid の**Column** セルの条件付き書式用のKnockoutJSバインディング。

wjStyle バインディングは、列のセルの条件付き書式を提供するために**wjFlexGridColumn** バインディングと組み合わせて使用します。 次に例を示します。

```
<div data-bind="wjFlexGridColumn:{
        header:'Change',
        binding:'changePercent',
        format:'p2',
        width:'*'
        },
        wjStyle:{ color: getAmountColor($item().change) }"></div>
```

wjStyleで使用する構文は、KnockoutJSの組み込みの**style**バインディングと同じです。 ビューモデルのプロパティに加えて、以下のオブザーバブル変数もバインディング式で使用できます。

$item 現在の行にバインドされている項目を参照します。
$row 行インデックス。
$col 列インデックス。

wjTooltip クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

Tooltip クラス用のKnockoutJSバインディング。

wjTooltip バインディングは、ページ上の要素にツールチップを追加する場合に使用します。 **wjTooltip** はHTMLコンテンツ、スマート配置、およびタッチをサポートします。

wjTooltip バインディングは、ツールチップを適用する対象の要素で指定します。値はツールチップのテキストまたはテキストを含む要素のIDです。次に例を示します。

```
<p data-bind="wjTooltip:'#fineprint'" >
    通常の段落の内容...</p>
...
<div id="fineprint" style="display:none" >
  <h3>重要</h3>
  <p>
    今四半期のデータは比例配分によって得られた推定値です...</p>
</div>
```

wjTreeView クラス

ファイル wijmo.knockout.js
モジュール **wijmo.knockout**
表示 ☐継承されたメンバー ☐イベント発生元

TreeView オブジェクトに使用されるKnockoutJS連結。 **wjTreeView** 連結を使用して、KnockoutJSアプリケーションに**TreeView** コントロールを追加できます。次に例を示します。   **wjTreeView**連結は、**TreeView** クラスのすべての読み取り/書き込み可能プロパティおよびイベントをサポートします。

wijmo/wijmo.angular2.directiveBase モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.directiveBase**

内部共通サービスとプラットフォームオプションを含むAngular 2モジュール用のBasic Wijmo。

wijmo.angular2.directiveBase wijmo.angular2.directiveBaseは、アンビエントモジュール名を使用してコードにインポートできる外部のTypeScriptモジュールです。次に例を示します。

```
import * as wjBase from 'wijmo/wijmo.angular2.directiveBase';

wjBase.WjOptions.asyncBindings = false;
```

クラス

- WjOptions

WjOptions クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.directiveBase**

WijmoのAngular 2相互運用のグローバルオプションを公開します。

プロパティ

- asyncBindings

プロパティ

- STATIC asyncBindings

Wijmoコンポーネントが双方向バインドプロパティのバインディングソースを非同期的に更新するかまたは同期的に更新するかを示します。 このプロパティがtrue（デフォルト）に設定されている場合、双方向バインディングのあるWijmoコンポーネントのプロパティ（WjInputNumber.valueなど）を変更すると、現在の変更検出サイクルが完了した後でコンポーネントがバインディングソースプロパティを非同期に更新します。このプロパティーをfalseに設定すると、バインディングソースはすぐに更新されます。また、バインディングソースが更新された後に、このプロパティ値に応じて、該当するプロパティ変更イベント（WjInputNumber.valueChangedなど）も非同期的または同期的に発生されます。

コンポーネントの**asyncBindings** プロパティに特定のブール値を設定することでこのグローバル設定を、Wijmoコンポーネントの特定のインスタンスに対して変更できます。

非同期的なバインディングソースの更新は、Wijmoのバージョン350で紹介されました。以前のバージョンでは、コンポーネントのプロパティが変更された直後にバインディングソースが更新されていました。この場合、デバッグモードでAngularを実行しているアプリケーションで**ExpressionChangedAfterItHasBeenCheckedError** 例外が発生する可能性があります。 たとえば、コンポーネントのプロパティ値が0.12345に設定されている場合、 **format**プロパティを**'n2'** に設定して**WjInputNumber** コンポーネントの**value** プロパティに双方向バインドすると、WjInputNumberは即座に値を0.12に変換します。これにより、コンポーネントのプロパティ（バインディング元）がAngularによって更新され、その値が0.12345から0.12に変更されます。このソース更新が同期的に実行される場合、バインディングソースプロパティが同じ変更検出サイクルで値を変更しますが、これはAngularによって禁止されています。 Angularがデバッグモードで実行されている場合、 変更検出サイクルごとに特別なチェックが実行されます。そののとき、変更が検出され、**ExpressionChangedAfterItHasBeenCheckedError** 例外が発生します。 現在の変更検出サイクルが完了した後にバインディングソースプロパティが更新されるため、非同期てきなバインディングソースの更新によってこの問題が解決されます。

ExpressionChangedAfterItHasBeenCheckedError が問題ではなく、バインド元の更新が行われた際にアプリケーションロジックの一部が敏感な場合は、グローバル**asyncBindings** プロパティをfalseに設定してこの機能を変更できます。ただし、これは、最初のWijmoコンポーネントがアプリケーションロジックによってインスタンス化される前に実行する必要があります。これを行う最も良い場所は、アプリケーションのroot NgModuleを宣言しているファイルとなります。 上記は次のようなコードで行うことができます：

```
import * as wjBase from 'wijmo/wijmo.angular2.directiveBase';
wjBase.WjOptions.asyncBindings = false;
```

または、**asyncBindings** プロパティを使用して、特定のコンポーネントの更新モードを変更することもできます。 次に例を示しています。

```
<wj-input-number [asyncBindings]="false" [(value)]="amount"></wj-input-number>
```

型 **boolean**

wijmo/wijmo.angular2.core モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.core**

wijmoモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.coreは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjCore from 'wijmo/wijmo.angular2.core';

@Component({
    directives: [wjCore.WjTooltip],
    template: '<span [wjTooltip]="'Greeting'">Hello</span>',
    selector: 'my-cmp',
})
export class MyCmp {
}
```

クラス

- WjComponentLoader
- WjTooltip

WjComponentLoader クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.core**

（作成中）

WjTooltip クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.core**

Tooltip クラスに対応するAngular 2ディレクティブ。

wjTooltipディレクティブを使用して、ページ上の要素にツールチップを追加できます。 wjTooltipディレクティブは、HTMLコンテンツ、スマート配置、およびタッチをサポートします。

wjTooltipディレクティブは、ツールチップの適用先の 要素に追加されるパラメータとして指定します。パラメータの値は、 ツールチップのテキスト、またはテキストを含む要素のIDです。次に例を示します。

```
<p [wjTooltip]="'#fineprint'" >
    通常の段落コンテンツ...</p>
...
<div id="fineprint" style="display:none">
  <h3>重要</h3>
  <p>
    今四半期のデータは比例配分によって得られた
    推定値です。</p>
</div>
```

プロパティ

- initialized
- isInitialized

メソッド

- created

プロパティ

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

wijmo/wijmo.angular2.input モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.input**

wijmo.inputモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.inputは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjInput from 'wijmo/wijmo.angular2.input';

@Component({
    directives: [wjInput.WjInputNumber],
    template: '<wj-input-number [(value)]="amount"></wj-input-number>',
    selector: 'my-cmp',
})
export class MyCmp {
    amount = 0;
}
```

クラス

- WjAutoComplete
- WjCalendar
- WjCollectionViewNavigator
- WjCollectionViewPager
- WjColorPicker
- WjComboBox
- WjContextMenu
- WjInputColor
- WjInputDate
- WjInputDateTime
- WjInputMask
- WjInputNumber
- WjInputTime
- WjItemTemplate
- WjListBox
- WjMenu
- WjMenuItem
- WjMenuSeparator
- WjMultiAutoComplete
- WjMultiSelect
- WjPopup

WjAutoComplete クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **AutoComplete**
表示 ☐継承されたメンバー ☐イベント発生元

AutoComplete コントロールに対応するAngular 2コンポーネント。

wj-auto-completeコンポーネントを使用して、Angular 2アプリケーションに**AutoComplete**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjAutoCompleteコンポーネントは、**AutoComplete**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoExpandSelection
- collectionView
- controlTemplate
- cssMatch
- delay
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceFunction
- listBox
- lostFocusNg

- maxDropDownHeight
- maxDropDownWidth
- maxItems
- minLength
- placeholder
- rightToLeft
- searchMemberPath
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- text
- textChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?: any): AutoComplete
```

AutoComplete クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **AutoComplete**
戻り値 **AutoComplete**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

cssMatch

検索語と一致するすべての部分をコンテンツで強調表示するために使用する CSSクラスの名前を取得または設定します。

継承元 **AutoComplete**
型 **string**

delay

キーストロークが発生してから検索が実行されるまでの遅延（ミリ秒単位）を取得または設定します。

The default value for this property is **500** milliseconds.

継承元 **AutoComplete**
型 **number**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 **ComboBox**
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **DropDown**
型 **HTMLInputElement**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDroppedDownChangedNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isDroppedDownChangingNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **ComboBox** control, and to true on the **AutoComplete** control.

継承元 **ComboBox**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

itemsSourceFunction

ユーザーの入力に従ってリスト項目を動的に提供する関数を取得または設定します。

この関数は以下の3つのパラメーターをとります。

- ユーザーが入力したクエリー文字列
- 返す項目の最大数
- 結果が取得されたときに呼び出すコールバック関数

例:

```
autoComplete.itemsSourceFunction = function (query, max, callback) {

  // サーバーから結果を取得します。
  var params = { query: query, max: max };
  $.getJSON('companycatalog.ashx', params, function (response) {

    // コントロールに結果を返します。
    callback(response);
  });
};
```

継承元 **AutoComplete**
型 **Function**

listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

maxItems

ドロップダウンリストに表示する項目の最大数を取得または設定します。

The default value for this property is **6**.

継承元 **AutoComplete**
型 **number**

minLength

オートコンプリート候補を検索するために必要な入力の最小長さを取得または設定します。

The default value for this property is **2**.

継承元 **AutoComplete**
型 **number**

placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

searchMemberPath

項目の検索時に使用するプロパティのカンマ区切りリストを含む文字列を取得または設定します。

デフォルトでは、**AutoComplete** コントロールは、**displayMemberPath** プロパティで指定された プロパティと比較して一致を検索します。**searchMemberPath** プロパティを使用すると、 追加のプロパティを使用して検索を行うことができます。

たとえば、以下のコードは、会社名を表示し、 会社名、シンボル、および国に基づいて検索を行います。

```
var ac = new wijmo.input.AutoComplete('#autoComplete', {
  itemsSource: companies,
  displayMemberPath: 'name',
  searchMemberPath: 'symbol,country'
});
```

継承元 **AutoComplete**
型 **string**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

- selectedIndexChangedNg

プログラムによるアクセスに使用されるWijmo **selectedIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValue

`selectedValuePath` を使用して取得された**`selectedItem`** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValuePath

`selectedValue` を**`selectedItem`** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **`ListBox`** should include group header items to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as an **`itemsSource`**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'selectedValue'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

WjCalendar クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.input
基本クラス Calendar
表示 ☐継承されたメンバー ☐イベント発生元

Calendar コントロールに対応するAngular 2コンポーネント。

wj-calendarコンポーネントを使用して、Angular 2アプリケーションに**Calendar**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjCalendarコンポーネントは、**Calendar**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- controlTemplate
- displayMonth
- displayMonthChangedNg
- firstDayOfWeek
- formatDayHeaders
- formatDays
- formatItemNg
- formatMonths
- formatYear
- formatYearMonth
- gotFocusNg
- hostElement
- initialized
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- lostFocusNg
- max
- min
- monthView
- repeatButtons
- rightToLeft
- selectionMode
- showHeader
- showYearPicker
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onDisplayMonthChanged
- onFormatItem
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- displayMonthChanged
- formatItem
- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): Calendar
```

Calendar クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **Calendar**
戻り値 **Calendar**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

STATIC controlTemplate

Calendar コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **Calendar**
型 **any**

displayMonth

カレンダーに表示されている月を取得または設定します。

継承元 **Calendar**
型 **Date**

displayMonthChangedNg

プログラムによるアクセスに使用されるWijmo **displayMonthChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **displayMonthChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

firstDayOfWeek

週の最初の曜日（カレンダーの最初の列に表示される曜日）を表す値を取得または設定します。

このプロパティをnullに設定すると、現在のカルチャのデフォルトが使用されます。週の最初の曜日は、英語カルチャでは日曜日（0）であり、ほとんどのヨーロッパカルチャでは月曜日（1）です。

継承元 **Calendar**
型 **number**

formatDayHeaders

月表示で、曜日の上に表示されるヘッダーの書式を取得または設定します。

このプロパティのデフォルト値は**'ddd'**です。

継承元 `Calendar`
型 **string**

formatDays

月表示で、日の表示書式を取得または設定します。

このプロパティのデフォルト値は 'd 'です（'d'の後のスペースは、短い日付パターンを表す標準な書式'd'と解釈されないようにします）。

継承元 `Calendar`
型 **string**

formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

formatMonths

年表示で、月の表示書式を取得または設定します。

このプロパティのデフォルト値は**'MMM'**です。

継承元 `Calendar`
型 **string**

formatYear

年表示で、月の上に表示される年の書式を取得または設定します。

このプロパティのデフォルト値は**'yyyy'**です。

継承元 `Calendar`
型 **string**

formatYearMonth

月表示で、カレンダーの上に表示される月と年の書式を取得または設定します。

このプロパティのデフォルト値は**'y'**です。

継承元 `Calendar`
型 **string**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **Calendar**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

カレンダーの日付をカスタマイズするフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意の日付に任意の内容を追加できます。そのため、カレンダーの外観と動作を全面的にカスタマイズすることが可能です。

この関数は以下の2つのパラメーターをとります。

- 書式設定する日付
- 日付を表すHTML要素

以下のサンプルコードは週末を無効な状態で表示します。

```
calendar.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

継承元 **Calendar**
型 **Function**

itemValidator

日付が選択可能かどうかを決定するバリデーター関数を取得または設定します。

このバリデーター関数は、調べる日付を表す1つのパラメーターを受け取り、その日付が無効で選択できない場合、falseを返す必要があります。

以下のサンプルコードは、週末を無効な状態で表示し、ユーザーがそれらの日付を選択できないようにします。

```
calendar.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

継承元 **Calendar**
型 **Function**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- max

ユーザーがカレンダーで選択できる最も遅い日付を取得または設定します。

The default value for this property is **null**, which means no latest date is defined.

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Calendar**
型 **Date**

- min

ユーザーがカレンダーで選択できる最も早い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined.

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Calendar**
型 **Date**

- monthView

カレンダーに1か月と1年のどちらを表示するかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **Calendar**
型 **boolean**

- repeatButtons

カレンダーボタンがリピートボタン(押された状態で繰り返し実行するボタン)として動作するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **Calendar**
型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectionMode

ユーザーが日や月を選択できるか、またはどの値も選択できないかを示す値を取得または設定します。

The default value for this property is **DateSelectionMode.Day**.

継承元 **Calendar**
型 **DateSelectionMode**

- showHeader

現在の月とナビゲーションボタンを含むヘッダ領域をコントロールに表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 `Calendar`
型 **boolean**

- showYearPicker

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 `Calendar`
型 **boolean**

- value

現在選択されている日付を取得または設定します。

The default value for this property is the current date.

継承元 `Calendar`
型 **Date**

- valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onDisplayMonthChanged

```
onDisplayMonthChanged(e?: EventArgs): void
```

displayMonthChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Calendar**
戻り値 **void**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

継承元 **Calendar**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Calendar**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

displayMonthChanged

displayMonth プロパティが変更された後に発生します。

継承元 **Calendar**
引数 **EventArgs**

formatItem

カレンダーの日を表す要素が作成されたときに発生します。

このイベントを使用してカレンダーの項目を表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、**formatItem** イベントを使用して週末を無効な状態にし、 カレンダーにグレーで表示されるようにします。

```
// 日曜日と土曜日を無効にします。
calendar.formatItem.addHandler(function (s, e) {
  var day = e.data.getDay();
  if (day == 0 || day == 6) {
    wijmo.addClass(e.item, 'wj-state-disabled');
  }
});
```

継承元 **Calendar**
引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

継承元 **Calendar**
引数 **EventArgs**

WjCollectionViewNavigator クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**

ICollectionView のナビゲータ要素に対応するAngular 2コンポーネント。

wj-collection-view-navigatorコンポーネントを使用して、 **ICollectionView** 内の項目間のナビゲートに使用できる要素を追加できます。Angular 2マークアップの構文については、 「Angular 2 マークアップ構文 (static/angular2Markup.html)」を参照してください。次に例を示します。

```
<wj-collection-view-navigator
  [cv]="myCollectionView">
</wj-collection-view-navigator>
```

プロパティ

- initialized
- isInitialized
- wjModelProperty

メソッド

- created

プロパティ

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

WjCollectionViewPager クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**

ICollectionView のページャ要素に対応するAngular 2コンポーネント。

wj-collection-view-pagerコンポーネントを使用して、 ページ区切り付きの**ICollectionView** のページ間のナビゲートに使用できる要素を追加できます。 Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。次に例を示します。

```
<wj-collection-view-pager
  [cv]="myCollectionView">
</wj-collection-view-pager>
```

プロパティ

- initialized
- isInitialized
- wjModelProperty

メソッド

- created

プロパティ

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

WjColorPicker クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **ColorPicker**
表示 ☐継承されたメンバー ☐イベント発生元

ColorPicker コントロールに対応するAngular 2コンポーネント。

wj-color-pickerコンポーネントを使用して、Angular 2アプリケーションに**ColorPicker**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjColorPickerコンポーネントは、**ColorPicker**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- controlTemplate
- gotFocusNg
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- lostFocusNg
- palette
- rightToLeft
- showAlphaChannel
- showColorString
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll

- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): ColorPicker
```

ColorPicker クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **ColorPicker**
戻り値 **ColorPicker**

プロパティ

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- STATIC controlTemplate

ColorPicker コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **ColorPicker**
型 **any**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

palette

パレットの色を含む配列を取得または設定します。このパレットは10色で構成され、10個の文字列の配列によって表されます。最初の2色は通常、白と黒です。

継承元 **ColorPicker**
型 **string**[]

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

showAlphaChannel

ユーザーが**ColorPicker** を使用して色のアルファチャネル（透明度）を編集できるかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **ColorPicker**
型 **boolean**

showColorString

ColorPicker に現在の色の文字列表現を表示するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ColorPicker**
型 **boolean**

value

現在選択されている色を取得または設定します。 このプロパティのデフォルトは「white」です。

継承元 **ColorPicker**
型 **string**

valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ColorPicker**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

継承元 **ColorPicker**
引数 **EventArgs**

WjComboBox クラス

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.input**
基本クラス **ComboBox**
表示 ☐継承されたメンバー ☐イベント発生元

ComboBox コントロールに対応するAngular 2コンポーネント。

wj-combo-boxコンポーネントを使用して、Angular 2アプリケーションに**ComboBox**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjComboBoxコンポーネントは、**ComboBox**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-combo-box** コンポーネントには、 **WjItemTemplate** を子コンポーネントとして含めることができます。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoExpandSelection
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- lostFocusNg
- maxDropDownHeight
- maxDropDownWidth
- placeholder

- rightToLeft
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- text
- textChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged

- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?): ComboBox
```

ComboBox クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **ComboBox**
戻り値 **ComboBox**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 `ComboBox`
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 `DropDown`
型 **HTMLInputElement**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 `DropDown`
型 **boolean**

isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 `ComboBox`
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 `DropDown`
型 **boolean**

- isDroppedDownChangedNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isDroppedDownChangingNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the `ComboBox` control, and to true on the `AutoComplete` control.

継承元 `ComboBox`
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

- itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

- listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

- maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

- selectedIndexChangedNg

プログラムによるアクセスに使用されるWijmo **selectedIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValue

`selectedValuePath` を使用して取得された**`selectedItem`** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValuePath

`selectedValue` を**`selectedItem`** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **`ListBox`** should include group header items to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as an **`itemsSource`**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'selectedValue'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

WjContextMenu クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**

コンテキストメニューに対応するAngular 2ディレクティブ。

wjContextMenuディレクティブを使用して、ページ上の要素に コンテキストメニューを追加できます。wjContextMenuディレクティブは **wj-menu**コンポーネントを基にし、 ユーザーが要素に対してコンテキストメニューを要求したときに（通常は 右クリックしたとき）、ポップアップメニューを表示します。

wjContextMenuディレクティブは、コンテキストメニューの適用先の 要素に追加されるパラメータとして指定します。パラメータの値は、 **wj-menu**コンポーネントへの参照です。次に例を示します。

```
<!-- コンテキストメニュー付きの段落 -->
<p [wjContextMenu]="menu" >
 この段落にはコンテキストメニューがあります。</p>

<!-- コンテキストメニューを定義します（非表示、ID付き）-->
<wj-menu #menu style="display:none">
 <wj-menu-item [cmd]="cmdOpen" [cmdParam] ="1">開く...</wj-menu-item>
 <wj-menu-item [cmd]="cmdSave" [cmdParam]="2">保存</wj-menu-item>
 <wj-menu-item [cmd]="cmdSave" [cmdParam]="3">名前を付けて保存...</wj-menu-item>
 <wj-menu-item [cmd]="cmdNew" [cmdParam] ="4">新規作成...</wj-menu-item>
 <wj-menu-separator></wj-menu-separator>
 <wj-menu-item [cmd]="cmdExit" [cmdParam]="5">終了</wj-menu-item>
</wj-menu >
```

WjInputColor クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **InputColor**
表示 ☐継承されたメンバー ☐イベント発生元

InputColor コントロールに対応するAngular 2コンポーネント。

wj-input-colorコンポーネントを使用して、Angular 2アプリケーションに**InputColor**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjInputColorコンポーネントは、**InputColor**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoExpandSelection
- colorPicker
- controlTemplate
- dropDown
- dropDownCssClass
- gotFocusNg
- hostElement
- initialized
- inputElement
- isAnimated
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- lostFocusNg
- palette
- placeholder
- rightToLeft
- showAlphaChannel
- showDropDownButton
- text
- textChangedNg
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onRefreshed
- onRefreshing
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- refreshed
- refreshing
- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputColor
```

InputColor クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **InputColor**
戻り値 **InputColor**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

colorPicker

ドロップダウンに表示される**ColorPicker** コントロールへの参照を取得します。

継承元 **InputColor**
型 **ColorPicker**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

- dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **DropDown**
型 **HTMLInputElement**

- isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDroppedDownChangedNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isDroppedDownChangingNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- palette

パレットの色を含む配列を取得または設定します。このパレットは10色で構成され、10個の文字列の配列によって表されます。最初の2色は通常、白と黒です。

継承元 `InputColor`
型 **string**[]

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 `DropDown`
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- showAlphaChannel

ユーザーが`ColorPicker` を使用して色のアルファチャネル（透明度）を編集できるかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 `InputColor`
型 **boolean**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- text

コントロールに表示されるテキストを取得または設定します。

継承元 **InputColor**
型 **string**

- textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- value

現在の色を取得または設定します。

継承元 **InputColor**
型 **string**

- valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **InputColor**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

継承元 **InputColor**
引数 **EventArgs**

WjInputDate クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **InputDate**
表示 ☐継承されたメンバー ☐イベント発生元

InputDate コントロールに対応するAngular 2コンポーネント。

wj-input-dateコンポーネントを使用して、Angular 2アプリケーションに**InputDate**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjInputDateコンポーネントは、**InputDate**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoExpandSelection
- calendar
- controlTemplate
- dropDown
- dropDownCssClass
- format
- gotFocusNg
- hostElement
- initialized
- inputElement
- inputType
- isAnimated
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- lostFocusNg
- mask
- max
- min
- placeholder
- repeatButtons
- rightToLeft
- selectionMode
- showDropDownButton
- showYearPicker

- text
- textChangedNg
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onRefreshed
- onRefreshing
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- refreshed
- refreshing
- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputDate
```

InputDate クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **InputDate**
戻り値 **InputDate**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

calendar

ドロップダウンボックスに表示される**Calendar** コントロールへの参照を取得します。

継承元 **InputDate**
型 **Calendar**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

- dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

- format

選択された日付の表示に使用される書式を取得または設定します。

書式文字列は、.NET形式の **日付書式文字列 (http://msdn.microsoft.com/ja-JP/library/8kb3ddd4(v=vs.110).aspx)**として表されます。 The default value for this property is **d**, the culture-dependent short date pattern (e.g. 6/15/2020 in the US, 15/6/2020 in France, or 2020/6/15 in Japan).

継承元 **InputDate**
型 **string**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **InputDate**
型 **HTMLInputElement**

- inputType

コントロールによってホストされているHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは"tel"に設定されており、マイナス記号と小数点記号を含む数値キーパッドが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"または"text"に変更してみてください。

"number"タイプのような入力要素はChromeで選択を防ぐため、推奨されません。詳細については、http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome を参照してください。

継承元 **InputDate**
型 **string**

- isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDroppedDownChangedNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isDroppedDownChangingNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

ドロップダウンカレンダーの日付をカスタマイズするフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意の日付に任意の内容を追加できます。そのため、カレンダーの外観と動作を全面的にカスタマイズすることが可能です。

この関数は以下の2つのパラメーターをとります。

- 書式設定する日付
- 日付を表すHTML要素

以下のサンプルコードは週末を無効な状態で表示します。

```
inputDate.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

継承元 **InputDate**
型 **Function**

itemValidator

日付が選択可能かどうかを決定するバリデーター関数を取得または設定します。

このバリデーター関数は、調べる日付を表す1つのパラメーターを受け取り、その日付が無効で選択できない場合、falseを返す必要があります。

以下のサンプルコードは、週末の日付を選択できないようにします。

```
inputDate.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

継承元 **InputDate**
型 **Function**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

mask

編集中に使用するマスクを取得または設定します。 マスクの書式は、**InputMask** コントロールで使用される書式と同じです。 指定する場合、このマスクは**format** プロパティの値と互換性がある必要があります。 たとえば、'MM/dd/yyyy'と書式設定された 日付の入力にマスク'99/99/9999'を使用できます。

継承元 **InputDate**
型 **string**

- max

ユーザーが入力できる最も遅い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined. **min** および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputDate**
型 **Date**

- min

ユーザーが入力できる最も早い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined. **min** および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputDate**
型 **Date**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- repeatButtons

カレンダーボタンがリピートボタン(押された状態で繰り返し実行するボタン)として動作するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **InputDate**
型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectionMode

ユーザーが日や月を選択できるか、またはどの値も選択できないかを示す値を取得または設定します。

このプロパティは、ドロップダウンカレンダーの動作に影響しますが、 日付の表示に使用する書式には影響しません。 **selectionMode** を'Month'に設定した場合は、通常、 **format** プロパティを'MMM yyyy'などの日を含まない書式に 設定する必要があります。次に例を示します。

```
var inputDate = new wijmo.input.InputDate('#el, {
  selectionMode: 'Month',
  format: 'MMM yyyy'
});
```

The default value for this property is **DateSelectionMode.Day**.

継承元 **InputDate**
型 **DateSelectionMode**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showYearPicker

Gets or sets a value that determines whether the drop-down calendar should display a list of years when the user clicks the header element on the year calendar.

The default value for this property is **true**.

継承元 **InputDate**
型 **boolean**

- text

コントロールに表示されるテキストを取得または設定します。

継承元 **InputDate**
型 **string**

- textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- value

現在の日付を取得または設定します。

The default value for this property is the current date.

継承元 **InputDate**
型 **Date**

- valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能性があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **InputDate**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

継承元 **InputDate**
引数 **EventArgs**

WjInputDateTime クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **InputDateTime**
表示 ☐継承されたメンバー ☐イベント発生元

InputDateTime コントロールに対応するAngular 2コンポーネント。

wj-input-date-timeコンポーネントを使用して、Angular 2アプリケーションに**InputDateTime**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjInputDateTimeコンポーネントは、**InputDateTime**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoExpandSelection
- calendar
- controlTemplate
- dropDown
- dropDownCssClass
- format
- gotFocusNg
- hostElement
- initialized
- inputElement
- inputTime
- inputType
- isAnimated
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- lostFocusNg
- mask
- max
- min
- placeholder
- repeatButtons
- rightToLeft
- selectionMode

- showDropDownButton
- showYearPicker
- text
- textChangedNg
- timeFormat
- timeMax
- timeMin
- timeStep
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onRefreshed
- onRefreshing
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- refreshed
- refreshing
- textChanged

- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputDateTime
```

InputDateTime クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **InputDateTime**
戻り値 **InputDateTime**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

calendar

ドロップダウンボックスに表示される**Calendar** コントロールへの参照を取得します。

継承元 **InputDate**
型 **Calendar**

STATIC controlTemplate

InputDateTime コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **InputDateTime**
型 **any**

- dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

- dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

- format

選択された日付の表示に使用される書式を取得または設定します。

書式文字列は、.NET形式の **日付書式文字列 (http://msdn.microsoft.com/ja-JP/library/8kb3ddd4(v=vs.110).aspx)**として表されます。 The default value for this property is **d**, the culture-dependent short date pattern (e.g. 6/15/2020 in the US, 15/6/2020 in France, or 2020/6/15 in Japan).

継承元 **InputDate**
型 **string**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **InputDate**
型 **HTMLInputElement**

- inputTime

内部の**InputTime** コントロールへの参照を取得します。これにより、 コントロールの完全なオブジェクトモデルにアクセスできます。

継承元 **InputDateTime**
型 **InputTime**

- inputType

コントロールによってホストされているHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは"tel"に設定されており、マイナス記号と小数点記号を含む数値キーパッドが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"または"text"に変更してみてください。

"number"タイプのような入力要素はChromeで選択を防ぐため、推奨されません。詳細については、
http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome を参照してください。

継承元 **InputDate**
型 **string**

- isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDroppedDownChangedNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isDroppedDownChangingNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

ドロップダウンカレンダーの日付をカスタマイズするフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意の日付に任意の内容を追加できます。そのため、カレンダーの外観と動作を全面的にカスタマイズすることが可能です。

この関数は以下の2つのパラメーターをとります。

- 書式設定する日付
- 日付を表すHTML要素

以下のサンプルコードは週末を無効な状態で表示します。

```
inputDate.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

継承元 **InputDate**
型 **Function**

itemValidator

日付が選択可能かどうかを決定するバリデーター関数を取得または設定します。

このバリデーター関数は、調べる日付を表す1つのパラメーターを受け取り、その日付が無効で選択できない場合、falseを返す必要があります。

以下のサンプルコードは、週末の日付を選択できないようにします。

```
inputDate.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

継承元 **InputDate**
型 **Function**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

mask

編集中に使用するマスクを取得または設定します。 マスクの書式は、**InputMask** コントロールで使用される書式と同じです。 指定する場合、このマスクは**format** プロパティの値と互換性がある必要があります。 たとえば、'MM/dd/yyyy'と書式設定された 日付の入力にマスク'99/99/9999'を使用できます。

継承元 **InputDate**
型 **string**

- max

ユーザーが入力できる最も遅い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined. **min** および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputDate**
型 **Date**

- min

ユーザーが入力できる最も早い日付を取得または設定します。

The default value for this property is **null**, which means no earliest date is defined. **min** および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputDate**
型 **Date**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- repeatButtons

カレンダーボタンがリピートボタン(押された状態で繰り返し実行するボタン)として動作するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **InputDate**
型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectionMode

ユーザーが日や月を選択できるか、またはどの値も選択できないかを示す値を取得または設定します。

このプロパティは、ドロップダウンカレンダーの動作に影響しますが、 日付の表示に使用する書式には影響しません。 **selectionMode** を'Month'に設定した場合は、通常、 **format** プロパティを'MMM yyyy'などの日を含まない書式に 設定する必要があります。次に例を示します。

```
var inputDate = new wijmo.input.InputDate('#el, {
  selectionMode: 'Month',
  format: 'MMM yyyy'
});
```

The default value for this property is **DateSelectionMode.Day**.

継承元 **InputDate**
型 **DateSelectionMode**

showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

showYearPicker

Gets or sets a value that determines whether the drop-down calendar should display a list of years when the user clicks the header element on the year calendar.

The default value for this property is **true**.

継承元 **InputDate**
型 **boolean**

text

コントロールに表示されるテキストを取得または設定します。

継承元 **InputDate**
型 **string**

textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

timeFormat

ドロップダウンリスト内の時刻の表示に使用される書式を取得または設定します。

このプロパティは、コントロールの入力要素に表示される値には影響しません。 入力要素の値は、**format** プロパティを使用して書式設定されます。

書式文字列は、.NET形式の**時間書式文字列**として表されます。

継承元 **InputDateTime**
型 **string**

timeMax

ユーザーが入力できる最遅時間を取得または設定します。

継承元 **InputDateTime**
型 **Date**

timeMin

ユーザーが入力できる最早時間を取得または設定します。

継承元 **InputDateTime**
型 **Date**

- timeStep

時刻のドロップダウンリストのエントリ間の分数を取得または設定します。

継承元 **InputDateTime**
型 **number**

- value

現在の日付を取得または設定します。

The default value for this property is the current date.

継承元 **InputDate**
型 **Date**

- valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **InputDate**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

継承元 **InputDate**
引数 **EventArgs**

WjInputMask クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **InputMask**
表示 ☐継承されたメンバー ☐イベント発生元

InputMask コントロールに対応するAngular 2コンポーネント。

wj-input-maskコンポーネントを使用して、Angular 2アプリケーションに**InputMask**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjInputMaskコンポーネントは、**InputMask**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- controlTemplate
- gotFocusNg
- hostElement
- initialized
- inputElement
- isDisabled
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- lostFocusNg
- mask
- maskFull
- placeholder
- promptChar
- rawValue
- rightToLeft
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus

- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputMask
```

InputMask クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **InputMask**
戻り値 **InputMask**

プロパティ

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

STATIC controlTemplate

InputMask コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **InputMask**
型 **any**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **InputMask**
型 **HTMLInputElement**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 `InputMask`
型 **boolean**

- isRequired

コントロール値が空以外の文字列でなければならないかどうかを 示す値を取得または設定します。

The default value for this property is **true**.

継承元 `InputMask`
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- mask

ユーザーのキー入力に伴って入力を検証するために使用されるマスクを取得または設定します。このマスクは、`InputMask` トピックにリストされているマスク文字を使用した 文字列として定義されます。

継承元 `InputMask`
型 **string**

- maskFull

マスクに対して完全に文字が入力されたかどうかを示す値を取得します。

継承元 `InputMask`
型 **boolean**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **InputMask**
型 **string**

- promptChar

コントロール内の入力位置を示すために使用される記号を取得または設定します。

継承元 **InputMask**
型 **string**

- rawValue

コントロールの未加工の値を取得または設定します（マスクリテラルを除く）。コントロールの未加工の値には、プロンプトとリテラル文字は含まれません。たとえば、**mask** プロパティが"AA-9999"に設定されている場合に、ユーザーが値"AB-1234"を入力すると、**rawValue** プロパティはマスクに含まれるハイフンを除外して"AB1234"を返します。

継承元 **InputMask**
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- value

コントロールに現在表示されているテキストを取得または設定します。

継承元 **InputMask**
型 **string**

- valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **InputMask**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **InputMask**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

継承元 **InputMask**
引数 **EventArgs**

WjInputNumber クラス

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.input**
基本クラス **InputNumber**
表示 ☐継承されたメンバー ☐イベント発生元

InputNumber コントロールに対応するAngular 2コンポーネント。

wj-input-numberコンポーネントを使用して、Angular 2アプリケーションに**InputNumber**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjInputNumberコンポーネントは、**InputNumber**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- controlTemplate
- format
- gotFocusNg
- hostElement
- initialized
- inputElement
- inputType
- isDisabled
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- lostFocusNg
- max
- min
- placeholder
- repeatButtons
- rightToLeft
- showSpinner
- step
- text
- textChangedNg
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- clamp
- containsFocus

- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputNumber
```

InputNumber クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **InputNumber**
戻り値 **InputNumber**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

STATIC controlTemplate

InputNumber コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **InputNumber**
型 **any**

format

編集中の数値の表示に使用される書式を取得または設定します（**Globalize** を参照）。

この書式文字列は、.NETスタイルの**標準の数値書式文字列**として表されます。

継承元 **InputNumber**
型 **string**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **InputNumber**
型 **HTMLInputElement**

inputType

コントロールによってホストされているHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは"tel"に設定されており、マイナス記号と小数点記号を含む数値キーパッドが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"または"text"に変更してみてください。

"number"タイプのような入力要素はChromeで選択を防ぐため、推奨されません。詳細については、http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome を参照してください。

継承元 `InputNumber`
型 **string**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 `InputNumber`
型 **boolean**

isRequired

コントロールの値を必ず数値に設定しなければならないか、（コントロールの内容を削除することによって）値をnullに設定できるかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 `InputNumber`
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- max

ユーザーが入力できる最も大きい数値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputNumber**
型 **number**

- min

ユーザーが入力できる最も小さい数値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputNumber**
型 **number**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **InputNumber**
型 **string**

- repeatButtons

スピナーボタンがリピートボタン(押された状態で繰り返し実行するボタン)として動作するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **InputNumber**
型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- showSpinner

コントロールに値を増減するスピナーボタンを表示するかどうかを示す値を取得または設定します（ステッププロパティを０以外の値に設定する必要があります）。

The default value for this property is **true**.

継承元 **InputNumber**
型 **boolean**

- step

ユーザーがスピナーボタンを押したときに**value** プロパティを増減する量を取得または設定します。

継承元 **InputNumber**
型 **number**

- text

コントロールに表示されているテキストを取得または設定します。

継承元 **InputNumber**
型 **string**

- textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- value

コントロールの現在の値を取得または設定します。

継承元 **InputNumber**
型 **number**

- valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

clamp

```
clamp(value: number): number
```

min プロパティおよび**max** プロパティで定義された範囲内の値を返します。

パラメーター

- **value**: **number**
  クランプする値。

継承元 **InputNumber**
戻り値 **number**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター
- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **InputNumber**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **InputNumber**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **InputNumber**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **InputNumber**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

継承元 **InputNumber**
引数 **EventArgs**

WjInputTime クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **InputTime**
表示 ☐継承されたメンバー ☐イベント発生元

InputTime コントロールに対応するAngular 2コンポーネント。

wj-input-timeコンポーネントを使用して、Angular 2アプリケーションに**InputTime**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjInputTimeコンポーネントは、**InputTime**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoExpandSelection
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- format
- formatItem
- formatItemNg
- gotFocusNg
- headerPath
- hostElement
- initialized
- inputElement
- inputType
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- lostFocusNg
- mask
- max

- maxDropDownHeight
- maxDropDownWidth
- min
- placeholder
- rightToLeft
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- step
- text
- textChangedNg
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onTextChanged
- onValueChanged
- refresh
- refreshAll

- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- textChanged
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): InputTime
```

InputTime クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **InputTime**
戻り値 **InputTime**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

format

選択された時刻の表示に使用される書式を取得または設定します（**Globalize** を参照）。

書式文字列は、.NET形式の**時間書式文字列**として表されます。

継承元 **InputTime**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

- formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 **ComboBox**
型 **string**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **InputTime**
型 **HTMLInputElement**

- inputType

コントロールによってホストされているHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは"tel"に設定されており、マイナス記号と小数点記号を含む数値キーパッドが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"または"text"に変更してみてください。

"number"タイプのような入力要素はChromeで選択を防ぐため、推奨されません。詳細については、http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome を参照してください。

継承元 **InputTime**
型 **string**

- isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDroppedDownChangedNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isDroppedDownChangingNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **ComboBox** control, and to true on the **AutoComplete** control.

継承元 **ComboBox**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

mask

編集時に使用するマスクを取得または設定します。

マスクの書式は**InputMask** コントロールで使用される書式と同じです。

指定する場合、このマスクは**format** プロパティの値と互換性がある必要があります。たとえば、短い時刻（書式't'）の入力に マスク'99:99 >LL'を使用できます。

継承元 **InputTime**
型 **string**

- max

ユーザーが入力できる最遅時間を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputTime**
型 **Date**

- maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

- maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

- min

ユーザーが入力できる最早時間を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **InputTime**
型 **Date**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

- selectedIndexChangedNg

プログラムによるアクセスに使用されるWijmo **selectedIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValue

selectedValuePath を使用して取得された**selectedItem** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValuePath

selectedValue を**selectedItem** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **ListBox** should include group header items to delimit data groups.

Data groups are created by modifying the **groupDescriptions** property of the **ICollectionView** object used as an **itemsSource**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- step

ドロップダウンリストの項目の間隔（分数）を取得または設定します。

このプロパティのデフォルト値は15分です。 null、ゼロ、または負の値に設定すると、ドロップダウンが無効になります。

継承元 **InputTime**
型 **number**

- text

コントロールに表示されているテキストを取得または設定します。

継承元 **InputTime**
型 **string**

- textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- value

現在の入力時間を取得または設定します。

継承元 **InputTime**
型 **Date**

- valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **InputTime**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

valueChanged

ユーザーアクションの結果として、またはコードでの割り当てにより、 **value** プロパティの値が変化すると発生します。

継承元 **InputTime**
引数 **EventArgs**

WjItemTemplate クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**

コントロールに対応するAngular 2コンポーネント。

[wjItemTemplate]ディレクティブは、

WjListBox 、**WjMenu** 、**WjComboBox** 、または**WjMultiSelect** コンポーネントのいずれかに含める必要があります。

[wjItemTemplate]ディレクティブは、それをネストして含むコンポーネントの項目のテンプレートを定義します。 テンプレートには、Angular 2の連結およびディレクティブを含む任意のHTMLフラグメントを含めることができます。 ローカルの**item**、**itemIndex**、および**control**テンプレート変数をAngular 2連結で使用して、 データ項目、そのインデックス、およびオーナーコントロールを参照できます。次に例を示します。

```
<wj-list-box style="max-height:300px;width:250px;"
            [itemsSource]="musicians">
  <template wjItemTemplate let-item="item" let-itemIndex="itemIndex">
      {{itemIndex + 1}}. <b>{{item.name}}</b>
      <div *ngIf="item.photo">
          <img [src]="item.photo" height="100" />
          <br />
          <a href="https://www.google.com/#newwindow=1&q=The+Beatles+"
             target="_blank"
             style="color:red">go there!</a>
      </div>
  </template>
</wj-list-box>
```

プロパティ

- initialized
- isInitialized

メソッド

- created

プロパティ

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

WjListBox クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **ListBox**
表示 ☐継承されたメンバー ☐イベント発生元

ListBox コントロールに対応するAngular 2コンポーネント。

wj-list-boxコンポーネントを使用して、Angular 2アプリケーションに**ListBox**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjListBoxコンポーネントは、**ListBox**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-list-boxコンポーネントには、**WjItemTemplate** 子ディレクティブを含めることができます。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- checkedItems
- checkedItemsChangedNg
- checkedMemberPath
- collectionView
- displayMemberPath
- formatItemNg
- gotFocusNg
- hostElement
- initialized
- isContentHtml
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemCheckedNg
- itemFormatter
- itemRole
- itemsChangedNg
- itemsSource
- lostFocusNg
- maxHeight
- rightToLeft
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showGroups
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getDisplayValue
- getItemChecked
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- isItemEnabled
- loadList
- onCheckedItemsChanged
- onFormatItem
- onGotFocus
- onItemChecked
- onItemsChanged
- onLoadedItems
- onLoadingItems
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- refresh
- refreshAll
- removeEventListener
- setItemChecked
- showSelection
- toggleItemChecked

イベント

- checkedItemsChanged
- formatItem
- gotFocus
- itemChecked
- itemsChanged
- loadedItems
- loadingItems
- lostFocus
- refreshed

- refreshing
- selectedIndexChanged

コンストラクタ

constructor

```
constructor(element: any, options?): ListBox
```

ListBox クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **ListBox**
戻り値 **ListBox**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

checkedItems

現在チェックされている項目を含む配列を取得または設定します。

継承元 **ListBox**
型 **any**[]

checkedItemsChangedNg

プログラムによるアクセスに使用されるWijmo **checkedItemsChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **checkedItemsChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

checkedMemberPath

各項目の横に配置されるCheckBoxの制御に使用されるプロパティの名前を 取得または設定します。このプロパティを使用して、複数選択ListBoxを作成します。項目をオンまたはオフにすると、**itemChecked** イベントが発生します。オン/オフにした項目を取得するには、**selectedItem** プロパティを使用します。現在オンの項目のリストを取得するには、**checkedItems** プロパティを使用します。

継承元 **ListBox**
型

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ListBox**
型 **ICollectionView**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ListBox**
型 **string**

formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isContentHtml

項目の内容がプレーンテキストとHTMLのどちらであるかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ListBox**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemCheckedNg

プログラムによるアクセスに使用されるWijmo **itemChecked**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemChecked** Wijmoイベント名を使用してください。

型 **EventEmitter**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
listBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ListBox**
型 **Function**

- itemRole

リスト項目に追加されている「role」属性の値を取得または設定します。このプロパティのデフォルト値は 「option」です。

継承元 **ListBox**
型 **string**

- itemsChangedNg

プログラムによるアクセスに使用されるWijmo **itemsChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemsChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- itemsSource

リスト項目を含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **ListBox**
型 **any**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- maxHeight

リストの最大の高さを取得または設定します。

継承元 **ListBox**
型 **number**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectedIndex

現在選択されている項目のインデックスを取得または設定します。

継承元 **ListBox**
型 **number**

- selectedIndexChangedNg

プログラムによるアクセスに使用されるWijmo **selectedIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectedItem

現在選択されている項目を取得または設定します。

継承元 **ListBox**
型 **any**

- selectedValue

`selectedValuePath` を使用して取得された**`selectedItem`** の値を取得または設定します。

継承元 **ListBox**
型 **any**

- selectedValuePath

`selectedValue` を**`selectedItem`** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ListBox**
型 **string**

- showGroups

Gets or sets a value that determines whether the **`ListBox`** should include group header items to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as an **`itemsSource`**.

The **`ListBox`** only shows the first level of grouping.

The default value for this property is **false**.

継承元 **ListBox**
型 **boolean**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'selectedValue'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index: number): string
```

指定したインデックスにある項目に対して表示されるテキストを（プレーンテキストとして）取得します。

パラメーター

- **index**: **number**
  itemsSource 内の項目のインデックス。

継承元 **ListBox**
戻り値 **string**

getDisplayValue

```
getDisplayValue(index: number): string
```

指定したインデックスにある項目に対して表示される文字列を取得します。この文字列は、**isContentHtml** プロパティの設定に応じてプレーンテキストまたはHTMLのどちらかになります。

パラメーター

- **index**: **number**
  itemsSource 内の項目のインデックス。

継承元 **ListBox**
戻り値 **string**

getItemChecked

```
getItemChecked(index: number): boolean
```

リストの項目のオン/オフ状態を取得します。このメソッドは、複数選択ListBoxにのみ適用できます （**checkedMemberPath** プロパティを参照）。

パラメーター

- **index**: **number**
  項目インデックス。

継承元 **ListBox**
戻り値 **boolean**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(e: HTMLElement): number
```

Gets the data index of an element within the list.

パラメーター

- **e**: **HTMLElement**
  検索する要素。

継承元 **ListBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isItemEnabled

```
isItemEnabled(index: number): void
```

指定したインデックスにある項目が有効かどうかを決定する値を取得します。

パラメーター

- **index**: **number**
  itemsSource 内の項目のインデックス。

継承元 **ListBox**
戻り値 **void**

loadList

```
loadList(): void
```

現在の**itemsSource** からの項目を含むリストをロードします。

継承元 **ListBox**
戻り値 **void**

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

checkedItemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ListBox**
戻り値 **void**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

継承元 **ListBox**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onItemChecked

```
onItemChecked(e?: EventArgs): void
```

itemChecked イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ListBox**
戻り値 **void**

onItemsChanged

```
onItemsChanged(e?: EventArgs): void
```

itemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ListBox**
戻り値 **void**

onLoadedItems

```
onLoadedItems(e?: EventArgs): void
```

loadedItems イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ListBox**
戻り値 **void**

onLoadingItems

```
onLoadingItems(e?: EventArgs): void
```

loadingItems イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ListBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ListBox**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **ListBox**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

setItemChecked

```
setItemChecked(index: number, checked: boolean): void
```

リストの項目のオン/オフ状態を設定します。このメソッドは、複数選択ListBoxにのみ適用できます （**checkedMemberPath** プロパティを参照）。

パラメーター

- **index**: **number**
  項目インデックス。
- **checked**: **boolean**
  項目の新しいチェック状態。

継承元 **ListBox**
戻り値 **void**

showSelection

```
showSelection(setFocus?: boolean): void
```

選択された項目を強調表示し、画面に入るようにスクロールします。

パラメーター

- **setFocus**: **boolean** OPTIONAL
  Whether to set the focus to the list after scrolling the selected item into view.

継承元 **ListBox**
戻り値 **void**

toggleItemChecked

```
toggleItemChecked(index: number): void
```

リストの項目のオン/オフ状態を切り替えます。このメソッドは、複数選択ListBoxにのみ適用できます （**checkedMemberPath** プロパティを参照）。

パラメーター

- **index**: **number**
  項目インデックス。

継承元 **ListBox**
戻り値 **void**

イベント

checkedItemsChanged

checkedItems プロパティの値が変更されたときに発生します。

継承元 **ListBox**
引数 **EventArgs**

formatItem

リスト項目を表す要素が作成されたときに発生します。このイベントを使用してリストの項目を表示用に書式設定できます。このイベントは、目的は**itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

継承元 **ListBox**
引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

itemChecked

ユーザーが現在の項目をオンまたはオフにすると発生します。このイベントは、 **checkedMemberPath** にプロパティの名前を設定して、コントロール内の各項目に チェックボックスを追加した場合に発生します。 オン/オフにした項目を取得するには、**selectedItem** プロパティを使用します。

継承元 **ListBox**
引数 **EventArgs**

itemsChanged

項目のリストが変更されたときに発生します。

継承元 **ListBox**
引数 **EventArgs**

loadedItems

リスト項目が生成される後に発生します。

継承元 **ListBox**
引数 **EventArgs**

loadingItems

リスト項目が生成される前に発生します。

継承元 **ListBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ListBox**
引数 **EventArgs**

WjMenu クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **Menu**
表示 ☐継承されたメンバー ☐イベント発生元

Menu コントロールに対応するAngular 2コンポーネント。

wj-menuコンポーネントを使用して、Angular 2アプリケーションに**Menu**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjMenuコンポーネントは、**Menu**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-menuコンポーネントには、次の子コンポーネントおよび子コンポーネントを含めることができます。 **WjMenuItem**、 **WjMenuSeparator**、と **WjItemTemplate**。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoExpandSelection
- collectionView
- command
- commandParameterPath
- commandPath
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- header
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isButton
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemClickedNg

- itemClickedNg
- itemFormatter
- itemsSource
- listBox
- lostFocusNg
- maxDropDownHeight
- maxDropDownWidth
- openOnHover
- owner
- placeholder
- rightToLeft
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- subItemsPath
- text
- textChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- hide
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemClicked
- onItemsSourceChanged
- onLostFocus
- onRefreshed

- onRefreshing
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll
- show

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemClicked
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?): Menu
```

Menu クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **Menu**
戻り値 **Menu**

プロパティ

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

command

項目がクリックされたときに実行されるコマンドを取得または設定します。

コマンドは以下の2つのメソッドを実装するオブジェクトです。

- **executeCommand(parameter)**: このメソッドはコマンドを実行します。
- **canExecuteCommand(parameter)**: このメソッドは、コントローラーがコマンドを実行できるかどうかを決定するブール値を返します。 このメソッドがfalseを返す場合、メニューオプションは無効になります。

また、**commandPath** プロパティを使用して個々の項目のコマンドを設定することもできます。

継承元 **Menu**
型 **any**

commandParameterPath

commandPath プロパティによって指定されたコマンドで使用するパラメーターを含むプロパティの名前を取得または設定します。

継承元 **Menu**
型 **string**

commandPath

ユーザーが項目をクリックしたときに実行されるコマンドを含むプロパティの名前を取得または設定します。

コマンドは以下の2つのメソッドを実装するオブジェクトです。

- **executeCommand(parameter)**: このメソッドはコマンドを実行します。
- **canExecuteCommand(parameter)**: このメソッドは、コントローラーがコマンドを実行できるかどうかを決定するブール値を返します。 このメソッドがfalseを返す場合、メニューオプションは無効になります。

継承元 **Menu**
型 **string**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

- displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

- dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

- dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

- formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

- formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- header

Menu 要素に表示されるHTMLテキストを取得または設定します。

継承元 **Menu**
型 **string**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 `ComboBox`
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 `DropDown`
型 **HTMLInputElement**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 `DropDown`
型 **boolean**

isButton

このMenu を通常のメニューではなく分割ボタンとして動作させるかどうかを 指定する値を取得または設定します。

通常のメニューと分割ボタンの違いは、ユーザーがメニューヘッダーを クリックしたときの動作です。 通常のメニューでは、ヘッダをクリックするとメニューオプションが表示されるか、非表示になります。 分割ボタンの場合は、ヘッダーをクリックすると、ドロップダウンリストから 項目を選択した場合と同様に、**itemClicked** イベントが発生したり、 最後に選択したオプションに関連するコマンドが呼び出されます。

メニュー項目のクリックと、分割ボタンのボタン部分のクリックを区別するには、 イベント送信元の**isDroppedDown** プロパティの値を確認します。 これがtrueの場合は、メニュー項目がクリックされました。 falseの場合は、ボタンがクリックされました。

たとえば、以下のコードは、デフォルトの項目/コマンドを変更するためにのみ ドロップダウンリストを使用し、ボタンがクリックされた場合にのみアクションを トリガする分割ボタンを実装します。

```
<-- view -->
<wj-menu is-button="true" header="Run" value="browser"
  item-clicked="itemClicked(s, e)">
  <wj-menu-item value="'Internet Explorer'">Internet Explorer</wj-menu-item>
  <wj-menu-item value="'Chrome'">Chrome</wj-menu-item>
  <wj-menu-item value="'FireFox'">FireFox</wj-menu-item>
  <wj-menu-item value="'Safari'">Safari</wj-menu-item>
  <wj-menu-item value="'Opera'">Opera</wj-menu-item>
</wj-menu>

// コントローラー
$scope.browser = 'Internet Explorer';
$scope.itemClicked = function (s, e) {

  // ドロップダウンでなかった場合は、ボタンがクリックされました
  if (!s.isDroppedDown) {
    alert('running ' + $scope.browser);
  }
}
```

継承元 **Menu**
型 **boolean**

isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isDroppedDownChangedNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isDroppedDownChangingNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **ComboBox** control, and to true on the **AutoComplete** control.

継承元 **ComboBox**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemClickedNg

プログラムによるアクセスに使用されるWijmo **itemClicked**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemClicked** Wijmoイベント名を使用してください。

型 **EventEmitter**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

- listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

- maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

- openOnHover

Gets or sets a value that determines whether the menu (and any sub-menus) should open and close automatically when the mouse hovers over the items.

The default value for this property is **false**.

継承元 **Menu**
型 **boolean**

- owner

Menu を所有する要素を取得または設定します。この変数は、単一のメニューはいくつかの要素のコンテキストメニューとして使用する場合wj-context-menuで設定します。

継承元 **Menu**
型 **HTMLElement**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 `ComboBox`
型 **number**

- selectedIndexChangedNg

プログラムによるアクセスに使用されるWijmo **selectedIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 `ComboBox`
型 **any**

- selectedValue

`selectedValuePath` を使用して取得された**`selectedItem`** の値を取得または設定します。

継承元 `ComboBox`
型 **any**

- selectedValuePath

`selectedValue` を**`selectedItem`** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 `ComboBox`
型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 `DropDown`
型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **ListBox** should include group header items to delimit data groups.

Data groups are created by modifying the **groupDescriptions** property of the **ICollectionView** object used as an **itemsSource**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- subItemsPath

Gets or sets the name of the property that contains an array with items to be displayed in a sub-menu.

継承元 **Menu**
型 **string**

- text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

- textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'selectedValue'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hide

```
hide(): void
```

メニューを非表示にします。このメソッドは、**show** メソッドを使用して表示したコンテキストメニューを非表示にする場合に便利です。

継承元 **Menu**
戻り値 **void**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。
- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemClicked

```
onItemClicked(e?: EventArgs): void
```

itemClicked イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Menu**
戻り値 **void**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

show

```
show(position?: any): void
```

指定された場所にメニューを表示します。

このメソッドは、メニューをコンテキストメニューとして使用して、 ページ内のいくつかの要素にアタッチする場合に便利です。次に例を示します。

```
// メニューを作成します
var div = document.createElement('div');
var menu = new wijmo.input.Menu(div, {
    itemsSource: 'New,Open,Save,Exit'.split(','),
    itemClicked: function (s, e) {
        alert('thanks for picking ' + menu.selectedIndex);
    }
});

// メニューをいくつかの要素のコンテキストメニューとして使用します
var element = document.getElementById('btn');
element.addEventListener('contextmenu', function (e) {
    e.preventDefault();
    menu.show(e);
});
```

パラメーター

- **position**: **any** OPTIONAL
  メニューを表示する位置を指定するオプションの**MouseEvent**または参照要素。

  指定しない場合、メニューは画面の中心に表示されます。

継承元 **Menu**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemClicked

ユーザーがメニューから項目を選択したときに発生します。イベントハンドラで、イベント送信元の **selectedIndex** プロパティを調べてどの項目が選択されたかを確認できます。

継承元 **Menu**
引数 **EventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

WjMenuItem クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
派生クラス **WjMenuSeparator**

コントロールに対応するAngular 2コンポーネント。

wj-menu-itemコンポーネントは、 **WjMenu** コンポーネントに含める必要があります。

wj-menu-itemコンポーネントを使用して、Angular 2アプリケーションにコントロールを追加できます。 Angular 2マークアップの構文については、「Angular 2マークアップ」を参照してください。

プロパティ

- initialized
- isInitialized
- wjProperty

メソッド

- created

プロパティ

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'itemsSource'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

WjMenuSeparator クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **WjMenuItem**
表示 ☐継承されたメンバー ☐イベント発生元

コントロールに対応するAngular 2コンポーネント。

wj-menu-separatorコンポーネントは、 **WjMenu** コンポーネントに含める必要があります。

wj-menu-separatorコンポーネントを使用して、Angular 2アプリケーションにコントロールを追加できます。 Angular 2マークアップの構文については、「Angular 2マークアップ」を参照してください。

プロパティ

- initialized
- isInitialized
- wjProperty

メソッド

- created

プロパティ

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

継承元 **WjMenuItem**
型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

継承元 **WjMenuItem**
型 **boolean**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'itemsSource'です。

継承元 **WjMenuItem**
型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

継承元 **WjMenuItem**
戻り値 **void**

WjMultiAutoComplete クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **MultiAutoComplete**
表示 ☐継承されたメンバー ☐イベント発生元

MultiAutoCompleteコントロールに対応するAngular 2コンポーネント。

wj-multi-auto-completeコンポーネントを使用して、Angular 2アプリケーションに**MultiAutoComplete**コントロールを追加できます。 Angular 2マークアップの構文については、 「Angular 2マークアップ」を参照してください。

WjMultiAutoCompleteコンポーネントは、**MultiAutoComplete**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoExpandSelection
- collectionView
- controlTemplate
- cssMatch
- delay
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceFunction
- listBox
- lostFocusNg

- maxDropDownHeight
- maxDropDownWidth
- maxItems
- maxSelectedItems
- minLength
- placeholder
- rightToLeft
- searchMemberPath
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedItems
- selectedItemsChangedNg
- selectedMemberPath
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- text
- textChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onSelectedItemsChanged

- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- selectedItemsChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?): MultiAutoComplete
```

MultiAutoComplete クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **MultiAutoComplete**
戻り値 **MultiAutoComplete**

プロパティ

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

cssMatch

検索語と一致するすべての部分をコンテンツで強調表示するために使用する CSSクラスの名前を取得または設定します。

継承元 **AutoComplete**
型 **string**

delay

キーストロークが発生してから検索が実行されるまでの遅延（ミリ秒単位）を取得または設定します。

The default value for this property is **500** milliseconds.

継承元 **AutoComplete**
型 **number**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 **ComboBox**
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 **DropDown**
型 **HTMLInputElement**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

isDroppedDownChangedNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

isDroppedDownChangingNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the **ComboBox** control, and to true on the **AutoComplete** control.

継承元 **ComboBox**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

itemsSourceFunction

ユーザーの入力に従ってリスト項目を動的に提供する関数を取得または設定します。

この関数は以下の3つのパラメーターをとります。

- ユーザーが入力したクエリー文字列
- 返す項目の最大数
- 結果が取得されたときに呼び出すコールバック関数

例:

```
autoComplete.itemsSourceFunction = function (query, max, callback) {

  // サーバーから結果を取得します。
  var params = { query: query, max: max };
  $.getJSON('companycatalog.ashx', params, function (response) {

    // コントロールに結果を返します。
    callback(response);
  });
};
```

継承元 **AutoComplete**
型 **Function**

- listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

- maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

- maxItems

ドロップダウンリストに表示する項目の最大数を取得または設定します。

The default value for this property is **6**.

継承元 **AutoComplete**
型 **number**

- maxSelectedItems

選択できる最大の項目数を取得または設定します。このプロパティをnull（デフォルト値）に設定すると、ユーザーは項目をいくつでも選択できます。

継承元 **MultiAutoComplete**
型 **number**

- minLength

オートコンプリート候補を検索するために必要な入力の最小長さを取得または設定します。

The default value for this property is **2**.

継承元 **AutoComplete**
型 **number**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- searchMemberPath

項目の検索時に使用するプロパティのカンマ区切りリストを含む文字列を取得または設定します。

デフォルトでは、**AutoComplete** コントロールは、**displayMemberPath** プロパティで指定された プロパティと比較して一致を検索します。**searchMemberPath** プロパティを使用すると、 追加のプロパティを使用して検索を行うことができます。

たとえば、以下のコードは、会社名を表示し、 会社名、シンボル、および国に基づいて検索を行います。

```
var ac = new wijmo.input.AutoComplete('#autoComplete', {
  itemsSource: companies,
  displayMemberPath: 'name',
  searchMemberPath: 'symbol,country'
});
```

継承元 **AutoComplete**
型 **string**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

- selectedIndexChangedNg

プログラムによるアクセスに使用されるWijmo **selectedIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

● selectedItems

現在選択されている項目が含まれる配列を取得または設定します。

継承元 **MultiAutoComplete**
型 **any**[]

● selectedItemsChangedNg

プログラムによるアクセスに使用されるWijmo **selectedItemsChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedItemsChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

● selectedMemberPath

どの項目が選択されるかの制御に使用されるプロパティの名前を取得または設定します。

継承元 **MultiAutoComplete**
型 **string**

● selectedValue

selectedValuePath を使用して取得された**selectedItem** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

● selectedValuePath

selectedValue を**selectedItem** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

● showDropDownButton

コントロールでドロップダウンボタンを表示しないことを示す値をオーバーライドします。

継承元 **MultiAutoComplete**
型 **boolean**

● showGroups

Gets or sets a value that determines whether the drop-down **ListBox** should include group header items to delimit data groups.

Data groups are created by modifying the **groupDescriptions** property of the **ICollectionView** object used as an **itemsSource**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'selectedItems'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onSelectedItemsChanged

```
onSelectedItemsChanged(e?: EventArgs): void
```

selectedItemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **MultiAutoComplete**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

selectedItemsChanged

selectedItems プロパティの値が変化すると発生します。

継承元 **MultiAutoComplete**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

WjMultiSelect クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **MultiSelect**
表示 ☐継承されたメンバー ☐イベント発生元

MultiSelect コントロールに対応するAngular 2コンポーネント。

wj-multi-selectコンポーネントを使用して、Angular 2アプリケーションに**MultiSelect**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjMultiSelectコンポーネントは、**MultiSelect**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-multi-select コンポーネントには、**WjItemTemplate** を子コンポーネントとして含めることができます。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoExpandSelection
- checkedItems
- checkedItemsChangedNg
- checkedMemberPath
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- headerFormat
- headerFormatter
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter

- itemsSource
- listBox
- lostFocusNg
- maxDropDownHeight
- maxDropDownWidth
- maxHeaderItems
- placeholder
- rightToLeft
- selectAllLabel
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showGroups
- showSelectAllCheckbox
- text
- textChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onCheckedItemsChanged
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- onTextChanged

- refresh
- refreshAll
- removeEventListener
- selectAll

イベント

- checkedItemsChanged
- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged
- textChanged

コンストラクタ

constructor

```
constructor(element: any, options?): MultiSelect
```

MultiSelect クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **MultiSelect**
戻り値 **MultiSelect**

プロパティ

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- autoExpandSelection

コントロールがクリックされたときに、選択範囲を自動的に単語/数字全体に拡張するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

checkedItems

現在チェックされている項目を含む配列を取得または設定します。

継承元 **MultiSelect**
型 **any**[]

checkedItemsChangedNg

プログラムによるアクセスに使用されるWijmo **checkedItemsChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **checkedItemsChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

checkedMemberPath

各項目の隣に配置したチェックボックスを制限するために使用されるプロパティの名前を取得または設定します。

継承元 **MultiSelect**
型 **string**

collectionView

項目ソースとして使用される **ICollectionView** オブジェクトを取得します。

継承元 **ComboBox**
型 **ICollectionView**

STATIC controlTemplate

DropDown コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **DropDown**
型 **any**

displayMemberPath

項目の視覚表示として使用するプロパティの名前を取得または設定します。

継承元 **ComboBox**
型 **string**

dropDown

isDroppedDown プロパティがtrueに設定されているときに表示されるドロップダウン要素を取得します。

継承元 **DropDown**
型 **HTMLElement**

dropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定します。

このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、 コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。

継承元 **DropDown**
型 **string**

formatItem

ドロップダウンリストの項目が作成されると発生するイベント。このイベントを使用して、リスト項目のHTMLを変更できます。詳細については、**formatItem** イベントを参照してください。

継承元 **ComboBox**
型 **Event**

formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

headerFormat

maxHeaderItems より以上の項目がチェックされた場合、ヘッダー コンテンツを作成するために使用する書式文字列を取得または設定します。書式文字列は、'{count}'の置換文字列が含まれて、現在にチェックされた項目数と置き換えます。英語カルチャでは、このプロパティのデフォルト値は'{count:n0} items selected'です。

継承元 **MultiSelect**
型 **string**

headerFormatter

コントロールのヘッダにHTMLを得る関数を取得または設定します。

デフォルトでは、コントロールのヘッダの内容は**placeholder** 、**maxHeaderItems** 、および現在の選択に 基づいて決定します。

アプリケーションの基準に基づいて、カスタム文字列を返す関数を指定すると、ヘッダ内容のカスタマイズができます。

継承元 **MultiSelect**
型 **Function**

headerPath

コントロールの入力要素に表示される値を取得するために使用するプロパティ名を取得または設定します。

このプロパティのデフォルト値はnullです。この場合、コントロールは、ドロップダウンリストの選択項目と同じ内容を入力要素に表示します。

入力要素に表示される値をドロップダウンリストに表示される値とは切り離す場合は、このプロパティを使用します。たとえば、入力要素には項目名を表示し、ドロップダウンリストには追加情報を表示することができます。

継承元 `ComboBox`
型 **string**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

inputElement

コントロールによってホストされているHTML入力要素を取得します。このプロパティは、入力要素の属性をカスタマイズする場合に使用します。

継承元 `DropDown`
型 **HTMLInputElement**

isAnimated

ドロップダウンを表示するときにコントロールがフェードインアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 `DropDown`
型 **boolean**

isContentHtml

ドロップダウンリストに項目をプレーンテキストとして表示するか、HTMLとして表示するかを 示す値を取得または設定します。

The default value for this property is **false**.

継承元 `ComboBox`
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

- isDroppedDown

ドロップダウンが現在表示されているかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 `DropDown`
型 **boolean**

- isDroppedDownChangedNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isDroppedDownChangingNg

プログラムによるアクセスに使用されるWijmo **isDroppedDownChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isDroppedDownChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

- isEditable

入力要素の内容をitemsSourceコレクション内の項目に制限するかどうかを決定する値を取得または設定します。

This property defaults to false on the `ComboBox` control, and to true on the `AutoComplete` control.

継承元 `ComboBox`
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してコントロール値を変更できるかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **DropDown**
型 **boolean**

- isRequired

コントロール値をnull以外の値に設定する必要があるか、それとも（コントロールのコンテンツを削除することで）nullに設定できるかを決定する値を取得または設定します。

This property defaults to true for most controls, including **ComboBox**, **InputDate**, **InputTime**, **InputDateTime**, and **InputColor**. It defaults to false for the **AutoComplete** control.

継承元 **DropDown**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- itemFormatter

リストに表示される値のカスタマイズに使用される関数を取得または設定します。 この関数は、2つの引数として項目インデックスとデフォルトのテキストまたはHTMLを 受け取り、表示する新しいテキストまたはHTMLを返します。

書式設定関数がスコープ（意味のある'this'値など）を必要とする場合は、 'bind'関数を使用してフィルタを設定し、 'this'オブジェクトを指定してください。次に例を示します。

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

継承元 **ComboBox**
型 **Function**

- itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Setting this property to an array causes the **ComboBox** to create an internal **ICollectionView** object that is exposed by the **collectionView** property.

The **ComboBox** selection is determined by the current item in its **collectionView**. By default, this is the first item in the collection. You may change this behavior by setting the **currentItem** property of the **collectionView** to null.

継承元 **ComboBox**
型 **any**

- listBox

ドロップダウンに示されている **ListBox** コントロールを取得します。

継承元 **ComboBox**
型 **ListBox**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- maxDropDownHeight

ドロップダウンリストの最大の高さを取得または設定します。

継承元 **ComboBox**
型 **number**

- maxDropDownWidth

ドロップダウンリストの最大の幅を取得または設定します。

ドロップダウンリストの幅は、コントロール自体の幅によっても制限されます（その値はドロップダウンの最小幅を表します）。

継承元 **ComboBox**
型 **number**

- maxHeaderItems

コントロールのヘッダに表示する項目の最大数を取得または設定します。

項目が選択されていない場合、ヘッダが、**placeholder** プロパティで 指定されたテキストが表示されます。

選択された項目の数は、**maxHeaderItems** プロパティの値より以下に ある場合、選択された項目はヘッダに示します。

選択された項目の数は、**maxHeaderItems** プロパティの値より大きい場合、 ヘッダは選択された項目ではなく、項目数表示します。

継承元 **MultiSelect**
型 **number**

- placeholder

コントロールが空のときにヒントとして表示される文字列を取得または設定します。

継承元 **DropDown**
型 **string**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectAllLabel

showSelectAllCheckbox プロパティがtrueに設定されている場合、表示される「すべて選択」チェックボックスのラベルとして使用する文字列を取得または設定します。

このプロパティはデフォルトでnullに設定されます。これにより、コントロールには、ローカライズ化された文字列 「すべて選択」が表示されます。

継承元 **MultiSelect**
型 **string**

- selectedIndex

ドロップダウンリストで現在選択されている項目のインデックスを取得または設定します。

継承元 **ComboBox**
型 **number**

- selectedIndexChangedNg

プログラムによるアクセスに使用されるWijmo **selectedIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectedItem

ドロップダウンリストで現在選択されている項目を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValue

selectedValuePath を使用して取得された**selectedItem** の値を取得または設定します。

継承元 **ComboBox**
型 **any**

- selectedValuePath

selectedValue を**selectedItem** から取得するために使用するプロパティの名前を 取得または設定します。

継承元 **ComboBox**
型 **string**

- showDropDownButton

コントロールにドロップダウンボタンを表示するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **DropDown**
型 **boolean**

- showGroups

Gets or sets a value that determines whether the drop-down **ListBox** should include group header items to delimit data groups.

Data groups are created by modifying the **groupDescriptions** property of the **ICollectionView** object used as an **itemsSource**.

The default value for this property is **false**.

継承元 **ComboBox**
型 **boolean**

- showSelectAllCheckbox

コントロールのすべての項目を選択または選択解除するために、項目の上に「すべて選択」チェックボックスを表示するかどうかを取得または設定します。

The default value for this property is **false**.

継承元 **MultiSelect**
型 **boolean**

- text

コントロールに表示されるテキストを取得または設定します。

継承元 **DropDown**
型 **string**

- textChangedNg

プログラムによるアクセスに使用されるWijmo **textChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **textChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'checkedItems'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能性があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDisplayText

```
getDisplayText(index?: number): string
```

指定したインデックスにある項目に対して表示される文字列を（プレーンテキストとして）取得します。

パラメーター

- **index**: **number** OPTIONAL
  テキストを取得する項目のインデックス。

継承元 **ComboBox**
戻り値 **string**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

指定した文字列と一致する最初の項目のインデックスを取得します。

パラメーター

- **text**: **string**
  検索するテキスト。

- **fullMatch**: **boolean**
  完全一致で検索するか、前方一致で検索するか。

継承元 **ComboBox**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

checkedItemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **MultiSelect**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

isDroppedDownChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

isDroppedDownChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **DropDown**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **ComboBox**
戻り値 **void**

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

textChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **DropDown**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

selectAll

```
selectAll(): void
```

コントロールにフォーカスを設定してそのすべての内容を選択します。

継承元 **DropDown**
戻り値 **void**

イベント

checkedItemsChanged

checkedItems プロパティの値が変更されたときに発生します。

継承元 **MultiSelect**
引数 **EventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

isDroppedDownChanged

ドロップダウンが表示または非表示になった後に発生します。

継承元 **DropDown**
引数 **EventArgs**

isDroppedDownChanging

ドロップダウンが表示または非表示になる前に発生します。

継承元 **DropDown**
引数 **CancelEventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **ComboBox**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **ComboBox**
引数 **EventArgs**

textChanged

text プロパティの値が変更されたときに発生します。

継承元 **DropDown**
引数 **EventArgs**

WjPopup クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.input**
基本クラス **Popup**
表示 ☐継承されたメンバー ☐イベント発生元

Popup コントロールに対応するAngular 2コンポーネント。

wj-popupコンポーネントを使用して、Angular 2アプリケーションに**Popup**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjPopupコンポーネントは、**Popup**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- content
- dialogResult
- dialogResultEnter
- fadeIn
- fadeOut
- gotFocusNg
- hiddenNg
- hideTrigger
- hidingNg
- hostElement
- initialized
- isDisabled
- isDraggable
- isInitialized
- isTouching
- isUpdating
- isVisible
- lostFocusNg
- modal
- owner
- removeOnHide
- rightToLeft
- showingNg
- shownNg
- showTrigger
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate

- beforeUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hide
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onHidden
- onHiding
- onLostFocus
- onRefreshed
- onRefreshing
- onShowing
- onShown
- refresh
- refreshAll
- removeEventListener
- show

イベント

- gotFocus
- hidden
- hiding
- lostFocus
- refreshed
- refreshing
- showing
- shown

コンストラクタ

constructor

```
constructor(element: any, options?: any): Popup
```

Popup クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **Popup**
戻り値 **Popup**

プロパティ

content

このPopup に含まれるHTML要素を取得または設定します。

継承元 Popup
型 HTMLElement

dialogResult

Popup を非表示にした後に、そのコンテンツを処理するために使用できる値を取得または設定します。

このプロパティは、Popup が表示されたときにnullに設定され、ボタンクリックイベントに 応答して、または hide メソッドの呼び出しの中で設定できます。

継承元 Popup
型 any

dialogResultEnter

Popup 表示中にユーザーが［Enter］キーを押したときにdialogResult として 使用される値を取得または設定します。

ユーザーが［Enter］を押し、dialogResultEnter プロパティがnullでない場合、 ポップアップは、すべての子要素が有効な状態かどうかをチェックします。 有効な状態の場合は、ポップアップが閉じ、 dialogResult プロパティが dialogResultEnter プロパティの値に設定されます。

継承元 Popup
型 any

fadeIn

Popup を表示するときにフェードアウトアニメーションを使用する かどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 Popup
型 boolean

fadeOut

Popup を非表示にするときにフェードアウトアニメーションを使用する かどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 Popup
型 boolean

gotFocusNg

プログラムによるアクセスに使用されるWijmo gotFocusイベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の gotFocus Wijmoイベント名を使用してください。

型 EventEmitter

hiddenNg

プログラムによるアクセスに使用されるWijmo **hidden**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **hidden** Wijmoイベント名を使用してください。プログラムによるアクセスに使用されるWijmo **shown**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **shown** Wijmoイベント名を使用してください。

型 **EventEmitter**

hideTrigger

ポップアップを非表示にするアクションを取得または設定します。

デフォルトでは、**showTrigger** プロパティは**Blur** に設定されており、 フォーカスを失った場合、またはEscキーを押した時にポップアップが非表示になります。

hideTrigger プロパティを**Click** に設定すると、ポップアップはオーナ要素を クリックする場合、またはEscキーを押した時非表示になります。

hideTrigger プロパティを**None** に設定すると、Popupは**hide** メソッドを 呼び出します。

継承元 **Popup**
型 **PopupTrigger**

hidingNg

プログラムによるアクセスに使用されるWijmo **hiding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **hiding** Wijmoイベント名を使用してください。

型 **EventEmitter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isDraggable

ポップアップをそのヘッダーでマウスによってドラッグできるかどうかを示す値を取得または設定します。

ヘッダーは '.wj-dialog-header' のCSSセレクターによって識別されます。 ダイアログに 'wj-dialog-header'クラスの要素が含まれていない場合、ユーザーはポップアップをドラッグできません。

ポップアップをドラッグ可能にする場合は、 '.wj-dialog-header' CSSセレクタのcursorプロパティを設定することができます。次に例を示しています。

```
<style>
  .wj-popup {
    width: 30%;
  }
  .wj-dialog-header {
    cursor: move;
  }
</style>
```

The default value for this property is **false**.

継承元 **Popup**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isVisible

Popupが現在表示されているかどうかを決定する値を取得します。

継承元 **Popup**
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

modal

Popup をモーダルダイアログとして表示するかどうかを決定する値を 取得または設定します。

モーダルダイアログは、 **Popup** がページ内の他のコンテンツより目立つように、 暗い背景が付けられます。

ダイアログを完全にモーダルにするには、 **hideTrigger** プロパティを **None** に設定して、ユーザーが背景をクリックしても ダイアログを閉じることができないようにします。この場合は、**hide** メソッドが呼び出されるか、 ユーザーが［Esc］キーを押した場合にのみ、ダイアログが閉じられます。

The default value for this property is **false**.

継承元 **Popup**
型 **boolean**

owner

Popup を所有する要素を取得または設定します。オーナーがnullの場合、ポップアップはダイアログとして動作します。ダイアログは、画面の中央に配置され、**show** メソッドを使用して表示する必要があります。

継承元 **Popup**
型 **HTMLElement**

removeOnHide

Popup が非表示になっているときに**Popup** 要素をDOMから削除するか非表示にするかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **Popup**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

showingNg

プログラムによるアクセスに使用されるWijmo **showing**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **showing** Wijmoイベント名を使用してください。

型 **EventEmitter**

shownNg

プログラムによるアクセスに使用されるWijmo **shown**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **shown** Wijmoイベント名を使用してください。

型 **EventEmitter**

showTrigger

Popup を表示するアクションを取得または設定します。デフォルトでは、 **showTrigger** プロパティは**Click** に設定されており、オーナー要素をクリックすると、ポップアップが表示されます。 **showTrigger** プロパティを**None** に設定すると、ポップアップは**show** メソッドを呼び出すのみで表示されます。

継承元 **Popup**
型 **PopupTrigger**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は''です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hide

```
hide(dialogResult?: any): void
```

Popup を非表示にします。

パラメーター

- **dialogResult**: **any** OPTIONAL
  Popup を閉じる前に**dialogResult** プロパティに割り当てられる オプションの値。

継承元 **Popup**
戻り値 **void**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onHidden

```
onHidden(e?: EventArgs): void
```

hidden イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Popup**
戻り値 **void**

onHiding

```
onHiding(e: CancelEventArgs): boolean
```

hiding イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **Popup**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onShowing

```
onShowing(e: CancelEventArgs): boolean
```

showing イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**

継承元 **Popup**
戻り値 **boolean**

onShown

```
onShown(e?: EventArgs): void
```

shown イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Popup**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

show

```
show(modal?: boolean, handleResult?: Function): void
```

Popup を表示します。

パラメーター

- **modal**: **boolean** OPTIONAL
  ポップアップをモーダルダイアログとして表示するかどうか。 指定した場合、その値が **modalプロパティに設定されます**。
- **handleResult**: **Function** OPTIONAL
  ポップアップが非表示になると、コールバックが呼び出されます。 コールバックが指定されている場合は、パラメータとしてポップアップを受け取ります。

  handleResultコールバックを使用すると、**hidden** イベントにハンドラを アタッチしなくても、呼び出し元がモーダルダイアログの結果を処理できます。 たとえば、 以下のコードは、**CollectionView** 内の現在の項目を編集するための ダイアログを表示します。 編集結果は、**dialogResult** 値に応じてコミット またはキャンセルされます。 次に例を示します。

```
$scope.editCurrentItem = function () {
  $scope.data.editItem($scope.data.currentItem);
  $scope.itemEditor.show(true, function (e) {
    if (e.dialogResult == 'wj-hide-ok') {
      $scope.data.commitEdit();
    } else {
      $scope.data.cancelEdit();
    }
  });
}
```

継承元 **Popup**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

hidden

Popup が非表示になる後に発生します。

継承元 **Popup**
引数 **EventArgs**

hiding

Popup が非表示になる前に発生します。

継承元 **Popup**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

showing

Popup が表示される前に発生させます。

継承元 **Popup**
引数 **CancelEventArgs**

shown

ポップアップが表示された後に発生させます。

継承元 **Popup**
引数 **EventArgs**

wijmo/wijmo.angular2.grid モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.grid**

wijmo.gridモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.gridは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
<p>ここには、4つの列を持つデータバインドFlexGridコントロールがあります。</p>
<wj-flex-grid [itemsSource]="data">
  <wj-flex-grid-column
    [header]="'Country'"
    [binding]="'country'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Sales'"
    [binding]="'sales'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Expenses'"
    [binding]="'expenses'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Downloads'"
    [binding]="'downloads'">
  </wj-flex-grid-column>
</wj-flex-grid>
```

クラス

- WjFlexGrid
- WjFlexGridCellTemplate
- WjFlexGridColumn

列挙体

- CellTemplateType

WjFlexGrid クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid**
基本クラス **FlexGrid**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid コントロールに対応するAngular 2コンポーネント。

wj-flex-gridコンポーネントを使用して、Angular 2アプリケーションに**FlexGrid**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。次に例を示します。

```
<p>4つの列を持つデータ連結されたFlexGridコントロールの例:</p>
<wj-flex-grid [itemsSource]="data">
  <wj-flex-grid-column
    [header]="'Country'"
    [binding]="'country'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Sales'"
    [binding]="'sales'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Expenses'"
    [binding]="'expenses'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Downloads'"
    [binding]="'downloads'">
  </wj-flex-grid-column>
</wj-flex-grid>
```

WjFlexGridコンポーネントは、**FlexGrid**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

次のプロパティは、テンプレート連結には使用できません。 **scrollPosition**、 **selection** および **columnLayout** プロパティ。

wj-flex-gridコンポーネントには、以下の子コンポーネントを含めることができます: **WjFlexGridDetail** 、 **WjFlexGridFilter**、 **WjFlexGridColumn**、および **WjFlexGridCellTemplate**。

コンストラクタ

- constructor

プロパティ

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoScroll
- autoSearch
- autoSizedColumnNg
- autoSizedRowNg
- autoSizeMode
- autoSizingColumnNg
- autoSizingRowNg
- beginningEditNg
- bottomLeftCells

- cellEditEndedNg
- cellEditEndingNg
- cellFactory
- cells
- childItemsPath
- clientSize
- cloneFrozenCells
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- copiedNg
- copyingNg
- deferResizing
- deletedRowNg
- deletingRowNg
- draggedColumnNg
- draggedRowNg
- draggingColumnNg
- draggingColumnOverNg
- draggingRowNg
- draggingRowOverNg
- editableCollectionView
- editRange
- formatItemNg
- frozenColumns
- frozenRows
- gotFocusNg
- groupCollapsedChangedNg
- groupCollapsedChangingNg
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- initialized
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceChangedNg
- itemValidator
- keyActionEnter
- keyActionTab

- loadedRowsNg
- loadingRowsNg
- lostFocusNg
- mergeManager
- newRowAtTop
- pastedCellNg
- pastedNg
- pastingCellNg
- pastingNg
- prepareCellForEditNg
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- resizedColumnNg
- resizedRowNg
- resizingColumnNg
- resizingRowNg
- rightToLeft
- rowAddedNg
- rowEditEndedNg
- rowEditEndingNg
- rowEditStartedNg
- rowEditStartingNg
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollPositionChangedNg
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionChangedNg
- selectionChangingNg
- selectionMode
- showAlternatingRows
- showDropDown
- showErrors
- showGroups
- showMarquee
- showSelectedHeaders
- showSort
- sortedColumnNg
- sortingColumnNg
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- updatedLayoutNg

- updatedLayoutNg
- updatedViewNg
- updatingLayoutNg
- updatingViewNg
- validateEdits
- viewRange
- virtualizationThreshold
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseGroupsToLevel
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCopied
- onCopying

- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onItemsSourceChanging
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onRefreshed
- onRefreshing
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData

- setClipString
- startEditing
- toggleDropDownList

イベント

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- itemsSourceChanging
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- refreshed
- refreshing
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged

- selectionChanging
- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout
- updatingView

コンストラクタ

constructor

```
constructor(element: any, options?): FlexGrid
```

FlexGrid クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **FlexGrid**
戻り値 **FlexGrid**

プロパティ

activeEditor

現在アクティブになっているセルエディタを表す**HTMLInputElement**を取得します。

継承元 **FlexGrid**
型 **HTMLInputElement**

allowAddNew

ユーザーがソースコレクションに項目を追加できるようにグリッドに新規行テンプレートを表示するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、新規行テンプレートは表示されません。

継承元 **FlexGrid**
型 **boolean**

allowDelete

ユーザーが［Delete］キーを押したときにグリッドの選択されている行を削除するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、選択されている行は削除されません。

継承元 **FlexGrid**
型 **boolean**

allowDragging

ユーザーがマウスを使用して行、列、またはその両方をドラッグできるかどうかを決定する値を取得または設定します。

autoScroll プロパティがtrueに設定されている場合、ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされます。

グリッドでは、列のドラッグがデフォルトで有効になっています。

グリッドが連結モードでは、行をドラッグするには特別な 考慮が必要になります。

グリッドが連結モードでは、行をドラッグするとデータソースとの同期が失われます（たとえば行4は6行として参照されることがあります）。 これを回避するには、**draggedRow** イベントを処理し、データを新しい行の位置と同期させる必要があります。

また、**allowSorting** プロパティをfalseに設定する必要があります。そうしないと、行の順序がデータによって決定され、行をドラッグするのは無意味になります。

次のフィドルでは、連結グリッドでの行のドラッグを実行しています。 **Bound Row Dragging (http://jsfiddle.net/Wijmo5/kyg0qsda/)**.

継承元 **FlexGrid**
型 **AllowDragging**

allowMerging

グリッドのどの部分のセル結合を許可するかを取得または設定します。

継承元 **FlexGrid**
型 **AllowMerging**

allowResizing

ユーザーがマウスを使用して行、列、またはその両方をサイズ変更できるかどうかを決定する値を取得または設定します。

サイズ変更が可能な場合は、列ヘッダーセルの右端をドラッグして列を、行ヘッダーセルの下端をドラッグして行をサイズ変更できます。

ヘッダーセルの端をダブルクリックして、行および列をコンテンツに合わせて自動的にサイズ変更することもできます。 自動サイズ変更の動作は、**autoSizeMode** プロパティを使用してカスタマイズできます。

継承元 **FlexGrid**
型 **AllowResizing**

allowSorting

ユーザーが列ヘッダーセルをクリックして列をソートできるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

autoClipboard

グリッドがクリップボードショートカットを処理するかどうかを決定する値を取得または設定します。

次のクリップボードショートカットがあります。

［Ctrl］+［C］、［Ctrl］+［Ins］	グリッドの選択範囲をクリップボードにコピーします。
［Ctrl］+［V］、［Shift］+［Ins］	クリップボードのテキストをグリッドの選択範囲に貼り付けます。

クリップボード操作は、表示されている行および列だけが対象となります。

読み取り専用のセルは、貼り付け操作の影響を受けません。

継承元 **FlexGrid**
型 **boolean**

autoGenerateColumns

グリッドが**itemsSource** に基づいて列を自動的に生成するかどうかを 決定する値を取得または設定します。

列の生成は、少なくとも1項目を含む**itemsSource** プロパティに依存します。このデータ項目が検査され、列が作成されて、プリミティブ値（数値、文字列、Boolean、またはDate）を含むプロパティに連結されます。

プロパティをnullに設定すると、適切なタイプを推測する方法がないため、列は生成されません。このような場合は、**autoGenerateColumns** プロパティをfalseに設定し、明示的に列を作成する必要があります。次に例を示します。

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // データ項目にnull値が含まれる場合があります
  columns: [                  // そのため、列を明示的に定義します
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

継承元 **FlexGrid**
型 **boolean**

autoScroll

ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされるかどうかを決定する値を取得または設定します。

行と列のドラッグは、**allowDragging** プロパティによって制御されます。

このプロパティはデフォルトでtrueに設定されます。

継承元 **FlexGrid**
型 **boolean**

autoSearch

ユーザーが読み取り専用セルに入力するに伴ってグリッドでセルを検索するかどうかを決定する値を取得または設定します。

検索が現在選択されている列(編集不可能な場合)で行われます。検索は現在選択されている行から始まり、大文字と小文字を区別されません。

継承元 **FlexGrid**
型 **boolean**

autoSizedColumnNg

プログラムによるアクセスに使用されるWijmo **autoSizedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

autoSizedRowNg

プログラムによるアクセスに使用されるWijmo **autoSizedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

autoSizeMode

行または列のサイズを自動設定するときにどのセルを考慮に入れるかを取得または設定します。

このプロパティは、ユーザーが列ヘッダの端をダブルクリックしたときの動作を制御します。

デフォルトでは、その列のヘッダセルとデータセルの内容に基づいて列の幅が自動的に設定されます。このプロパティを使用すると、ヘッダのみまたはデータのみに基づいて列の幅を設定するように変更できます。

継承元 **FlexGrid**
型 **AutoSizeMode**

autoSizingColumnNg

プログラムによるアクセスに使用されるWijmo **autoSizingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

autoSizingRowNg

プログラムによるアクセスに使用されるWijmo **autoSizingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

beginningEditNg

プログラムによるアクセスに使用されるWijmo **beginningEdit**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **beginningEdit** Wijmoイベント名を使用してください。

型 **EventEmitter**

bottomLeftCells

左下のセルを含む**GridPanel** を取得します。

bottomLeftCells パネルは、行ヘッダの下、**columnFooters** パネルの左に表示されます。

継承元 **FlexGrid**
型 **GridPanel**

cellEditEndedNg

プログラムによるアクセスに使用されるWijmo **cellEditEnded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **cellEditEnded** Wijmoイベント名を使用してください。

型 **EventEmitter**

cellEditEndingNg

プログラムによるアクセスに使用されるWijmo **cellEditEnding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **cellEditEnding** Wijmoイベント名を使用してください。

型 **EventEmitter**

cellFactory

このグリッドのセルを作成および更新する**CellFactory** を取得または設定します。

継承元 **FlexGrid**
型 **CellFactory**

cells

データセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

childItemsPath

階層グリッドで子の行の生成に使用される 1 つ以上のプロパティの名前を取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `childItemsPath = 'items'` ）。

項目の異なるレベルに異なる名前を持つ子項目がある場合は、このプロパティに、各レベルの子項目を含むプロパティの名前から成る配列を設定します。（たとえば、 `childItemsPath = ['checks','earnings'];` ）。

サンプル

Show me (https://jsfiddle.net/Wijmo5/kk9j93bL)

継承元 **FlexGrid**
型 **any**

clientSize

コントロールのクライアントサイズを取得します（コントロールのサイズからヘッダーとスクロールバーを引いた値）。

継承元 **FlexGrid**
型 **Size**

cloneFrozenCells

スクロール中に知覚されるパフォーマンスを向上させるには、FlexGridがフリーズされたセルを複製し、別の要素に表示するかどうかを決定する値を取得または設定します。

このプロパティのデフォルト値はnullであり、ブラウザーによって一番適当な設定が選択されます。

継承元 **FlexGrid**
型 **boolean**

collectionView

グリッドデータを含む**ICollectionView** を取得します。

継承元 **FlexGrid**
型 **ICollectionView**

columnFooters

列フッターセルを保持する**GridPanel** を取得します。

columnFooters パネルは、グリッドセルの下、**bottomLeftCells** パネルの右に表示されます。これを使用して、グリッドデータの下にサマリー情報を表示できます。

以下の例では、**columnFooters** パネルに 1行を追加して、**aggregate** プロパティが設定された列に サマリーデータを表示する方法を示します。

```
function addFooterRow(flex) {

  // 集計を表示するGroupRowを作成します
  var row = new wijmo.grid.GroupRow();

  // 列フッターパネルに行を追加します
  flex.columnFooters.rows.push(row);

  // ヘッダーにシグマを表示します
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

継承元 **FlexGrid**
型 **GridPanel**

columnHeaders

列ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

columnLayout

現在の列レイアウトを定義するJSON文字列を取得または設定します。

列レイアウト文字列は、列とそのプロパティを含む配列を表します。これを使用して、ユーザーが定義した列レイアウトをセッションを越えて保持できます。また、ユーザーが列レイアウトを変更できるアプリケーションで元に戻す/やり直し機能を実装することもできます。

データマップはシリアル化できないため、列レイアウト文字列に**dataMap**　プロパティは含まれていません。

継承元　**FlexGrid**
型　**string**

columns

グリッドの列コレクションを取得します。

継承元　**FlexGrid**
型　**ColumnCollection**

controlRect

コントロールの外接矩形（ページ座標単位）を取得します。

継承元　**FlexGrid**
型　**Rect**

STATIC controlTemplate

FlexGrid コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元　**FlexGrid**
型　**any**

copiedNg

プログラムによるアクセスに使用されるWijmo **copied**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の**copied** Wijmoイベント名を使用してください。

型　**EventEmitter**

copyingNg

プログラムによるアクセスに使用されるWijmo **copying**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の**copying** Wijmoイベント名を使用してください。

型　**EventEmitter**

deferResizing

ユーザーがマウスボタンを放すまで、行および列のサイズ変更を遅らせるかどうかを決定する値を取得または設定します。

デフォルトでは、**deferResizing** はfalseに設定されており、ユーザーがマウスをドラッグするに伴って行および列がサイズ変更されます。このプロパティをtrueに設定すると、グリッドにサイズ変更マーカーが表示され、ユーザーがマウスボタンを放したときに初めて行または列のサイズが変更されます。

継承元　**FlexGrid**
型　**boolean**

deletedRowNg

プログラムによるアクセスに使用されるWijmo **deletedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **deletedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

deletingRowNg

プログラムによるアクセスに使用されるWijmo **deletingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **deletingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggedColumnNg

プログラムによるアクセスに使用されるWijmo **draggedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggedRowNg

プログラムによるアクセスに使用されるWijmo **draggedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingColumnNg

プログラムによるアクセスに使用されるWijmo **draggingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingColumnOverNg

プログラムによるアクセスに使用されるWijmo **draggingColumnOver**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingColumnOver** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingRowNg

プログラムによるアクセスに使用されるWijmo **draggingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

- draggingRowOverNg

プログラムによるアクセスに使用されるWijmo **draggingRowOver**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingRowOver** Wijmoイベント名を使用してください。

型 **EventEmitter**

- editableCollectionView

グリッドデータを含む**IEditableCollectionView** を取得します。

継承元 **FlexGrid**
型 **IEditableCollectionView**

- editRange

現在編集中のセルを識別する**CellRange** を取得します。

継承元 **FlexGrid**
型 **CellRange**

- formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

- frozenColumns

固定された列の数を取得または設定します。

固定された列は水平方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

- frozenRows

静止行の数を取得または設定します。

固定された行は垂直方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupCollapsedChangedNg

プログラムによるアクセスに使用されるWijmo **groupCollapsedChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **groupCollapsedChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupCollapsedChangingNg

プログラムによるアクセスに使用されるWijmo **groupCollapsedChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **groupCollapsedChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupHeaderFormat

グループヘッダーコンテンツを作成するために使用される書式文字列を取得または設定します。

この文字列は、任意のテキストと次の置換文字列を含むことができます。

- **{name}**：グループ化されるプロパティの名前。
- **{value}**：グループ化されるプロパティの値。
- **{level}**：グループレベル。
- **{count}**：このグループ内にある項目の合計数。

列がグループ化プロパティに連結されている場合は、列ヘッダーを使用して パラメータを置換し、列の書式とデータマップを使用して `{value}` パラメータを計算します。 列がない場合は、代わりにグループ情報が使用されます。

グループプロパティに連結された非表示の列を追加して、グループヘッダーセルの書式設定をカスタマイズすることもできます。

このプロパティのデフォルト値は
`'{name}: <b>{value}</b>({count:n0} items)'` です。 これは、 `'Country: UK (12 items)'` や
`'Country: Japan (8 items)'` のようなグループヘッダーを作成します。

継承元 **FlexGrid**
型 **string**

headersVisibility

行ヘッダおよび列ヘッダが表示されるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- imeEnabled

編集モードでないときに、グリッドがIME（Input Method Editor）をサポートするかどうかを決定する値を取得または設定します。

このプロパティは、日本語、中国語、韓国語など、IMEサポートを必要とする言語のサイト/アプリケーションにのみ関係します。

継承元 `FlexGrid`
型 **boolean**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してセル値を変更できるかどうかを判定する値を取得または設定します。

継承元 `FlexGrid`
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

itemFormatter

このグリッドのセルのカスタマイズに使用されるフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意のセルに任意の内容を追加できます。そのため、グリッドセルの外観と動作を非常に柔軟に制御することが可能です。

この関数は、セルを含む**GridPanel**、セルの行インデックスと列インデックス、セルを表すHTML要素の4つのパラメーターをとります。通常は、関数内でセル要素の**innerHTML**　プロパティを変更するようにします。

例:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == wijmo.grid.CellType.Cell) {

    // セルにスパークラインを描画します。
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

FlexGridはセルをリサイクルするため、**itemFormatter** によってセルのスタイル属性を変更する場合は、 セルの適切でないスタイル属性を事前にリセットする必要があります。例:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // カスタマイズする属性をリセットします。
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // このセルのcolorおよびbackgroundColor属性をカスタマイズします。
  ...
}
```

複数のクライアントがグリッドのレンダリングをカスタマイズできるようにする場合は（たとえば、ディレクティブや再利用可能なライブラリを作成するときなど）、代わりに**formatItem** イベントを使用することを検討してください。このイベントを使用すると、複数のクライアントがそれぞれ独自のハンドラをアタッチできます。

継承元　　**FlexGrid**
型　　**Function**

itemsSource

グリッドに表示される項目を含む配列または**ICollectionView** を取得または設定します。

継承元　　**FlexGrid**
型　　**any**

itemsSourceChangedNg

プログラムによるアクセスに使用されるWijmo **itemsSourceChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemsSourceChanged** Wijmoイベント名を使用してください。

型　　**EventEmitter**

itemValidator

セルに有効なデータが含まれているかどうかを決定するバリデータ関数を取得または設定します。

指定された場合、バリデータ関数はセルの行インデックスと列インデックスを含む2つのパラメータをとり、エラーの説明を含む文字列を返す必要があります。

このプロパティは、バインドされていないグリッドを処理する場合に特に便利です。バインドされたグリッドは、代わりに**getError** プロパティを使用して検証できます。

この例では、セルのすぐ上のセルと同じデータがセルに含まれないようにする方法を示します。

```
// 上のセルは、現在のセルと同じ値が含まれていないことを確認します
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'これは重複した値です...'
    }
  }
  return null; // エラーなし
}
```

継承元 **FlexGrid**
型 **Function**

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は**MoveDown** となり、コントロールがカーソルを次の行に移動します。 この動作は標準的なExcel式の動作です。

継承元 **FlexGrid**
型 **KeyAction**

keyActionTab

［Tab］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は、**None** となります。これにより、Tabキーが押されたときにブラウザ上で次または前のコントロールを選択できます。これは、ページのアクセシビリティを向上させるために推奨される設定になります。

以前のバージョンでは、デフォルトは**Cycle** に設定されていました。これにより、コントロールがカーソルを、グリッドを横切って下部に移動しました。これは標準的なExcelの動作ですが、アクセシビリティには適していません。

また、**CycleOut** の設定が存在します。これにより、カーソルがセルを通じて移動（**Cycle** ）してから、最後または最初のセルが選択されたときにページの次/前のコントロールに移動します 。

継承元 **FlexGrid**
型 **KeyAction**

loadedRowsNg

プログラムによるアクセスに使用されるWijmo **loadedRows**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadedRows** Wijmoイベント名を使用してください。

型 **EventEmitter**

loadingRowsNg

プログラムによるアクセスに使用されるWijmo **loadingRows**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadingRows** Wijmoイベント名を使用してください。

型 **EventEmitter**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

mergeManager

セルの結合方法を決定する**MergeManager** オブジェクトを取得または設定します。

継承元 **FlexGrid**
型 **MergeManager**

newRowAtTop

新しい行テンプレートをグリッドの上部に配置するか、下部に配置するかを示す値を取得または設定します。

newRowAtTop プロパティをtrueに設定した場合、新しい行テンプレートを常に表示したままにする場合は、**frozenRows** プロパティを 1に設定します。これにより、新しい行テンプレートは上部に固定され、ビューからスクロールオフされなくなります。

allowAddNew プロパティがtrueに設定されている場合、および**itemsSource** オブジェクトが新しい項目の追加をサポートしている場合にのみ、新しい行テンプレートが表示されます。

継承元 **FlexGrid**
型 **boolean**

pastedCellNg

プログラムによるアクセスに使用されるWijmo **pastedCell**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pastedCell** Wijmoイベント名を使用してください。

型 **EventEmitter**

pastedNg

プログラムによるアクセスに使用されるWijmo **pasted**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pasted** Wijmoイベント名を使用してください。

型 **EventEmitter**

pastingCellNg

プログラムによるアクセスに使用されるWijmo **pastingCell**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pastingCell** Wijmoイベント名を使用してください。

型 **EventEmitter**

pastingNg

プログラムによるアクセスに使用されるWijmo **pasting**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pasting** Wijmoイベント名を使用してください。

型 **EventEmitter**

prepareCellForEditNg

プログラムによるアクセスに使用されるWijmo **prepareCellForEdit**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **prepareCellForEdit** Wijmoイベント名を使用してください。

型 **EventEmitter**

preserveOutlineState

データがリフレッシュされたときに、グリッドがノードの展開/折りたたみ状態を保持するかどうかを決定する値を取得または設定します。

preserveOutlineState プロパティの実装は、JavaScriptの**Map** オブジェクトに基づいています。このオブジェクトはIE 9およびIE 10で利用できません。

継承元 **FlexGrid**
型 **boolean**

preserveSelectedState

データがリフレッシュされたときに、グリッドが行の選択状態を保持するかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

quickAutoSize

グリッドが列のサイズを自動調整するときに精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、自動サイズ変更の機能が無効になります。このプロパティをtrueに設定すると、各列の**quickAutoSize** プロパティの値に従って、その機能が有効になります。null（デフォルト値）に設定した場合、カスタムの **itemFormatter** を持たないグリッドの機能、または**itemFormatter** イベントにアタッチされたハンドラの機能が有効になります。

継承元 **FlexGrid**
型 **boolean**

resizedColumnNg

プログラムによるアクセスに使用されるWijmo **resizedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizedRowNg

プログラムによるアクセスに使用されるWijmo **resizedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizingColumnNg

プログラムによるアクセスに使用されるWijmo **resizingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizingRowNg

プログラムによるアクセスに使用されるWijmo **resizingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

rowAddedNg

プログラムによるアクセスに使用されるWijmo **rowAdded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowAdded** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditEndedNg

プログラムによるアクセスに使用されるWijmo **rowEditEnded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditEnded** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditEndingNg

プログラムによるアクセスに使用されるWijmo **rowEditEnding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditEnding** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditStartedNg

プログラムによるアクセスに使用されるWijmo **rowEditStarted**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditStarted** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditStartingNg

プログラムによるアクセスに使用されるWijmo **rowEditStarting**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditStarting** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowHeaderPath

行ヘッダーセルを作成するために使用されるプロパティの名前を取得または設定します。

行ヘッダーセルは表示されないし、選択することもできません。アクセシビリティツールと組み合わせて使用することを意図しています。

継承元 **FlexGrid**
型 **string**

rowHeaders

行ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

rows

グリッドの行コレクションを取得します。

継承元 **FlexGrid**
型 **RowCollection**

scrollPosition

グリッドのスクロールバーの値を表す**Point** を取得または設定します。

継承元 **FlexGrid**
型 **Point**

scrollPositionChangedNg

プログラムによるアクセスに使用されるWijmo **scrollPositionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **scrollPositionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

scrollSize

グリッド内容のサイズ（ピクセル単位）を取得します。

継承元 **FlexGrid**
型 **Size**

selectedItems

現在選択されているデータ項目を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは **selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selectedRows

現在選択されている行を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは **selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selection

現在の選択を取得または設定します。

継承元 **FlexGrid**
型 **CellRange**

selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionChangingNg

プログラムによるアクセスに使用されるWijmo **selectionChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectionMode

現在の選択モードを取得または設定します。

継承元 **FlexGrid**
型 **SelectionMode**

- showAlternatingRows

グリッドで交互表示行のセルに'wj-alt'クラスを追加するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、CSSを変更しなくても、交互表示行スタイルを無効にできます。

継承元 **FlexGrid**
型 **boolean**

- showDropDown

グリッドで、**showDropDown** プロパティがtrueに設定されている列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **FlexGrid**
型 **boolean**

- showErrors

グリッドで、検証エラーがあるセルとエラーの説明を含むツールチップに'wj-state-invalid' クラスを追加するかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**itemValidator** または**getError** プロパティを使用して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

- showGroups

データグループを区切るためにグリッドにグループ行を挿入するかどうかを決定する値を 取得または設定します。

データグループを作成するには、グリッドの **itemsSource** として使用される**ICollectionView** オブジェクトの**groupDescriptions** プロパティを変更します。

継承元 **FlexGrid**
型 **boolean**

- showMarquee

現在の選択範囲の周囲にマーキー要素を表示するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

- showSelectedHeaders

選択されているヘッダセルを示すためにクラス名を追加するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

- showSort

グリッドで、列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

ソートは、グリッドの**itemsSource** として使用される**ICollectionView** オブジェクトの**sortDescriptions** プロパティによって制御されます。

継承元 **FlexGrid**
型 **boolean**

- sortedColumnNg

プログラムによるアクセスに使用されるWijmo **sortedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **sortedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

- sortingColumnNg

プログラムによるアクセスに使用されるWijmo **sortingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **sortingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

- sortRowIndex

ソートインジケーターを表示する列ヘッダパネルの行のインデックスを取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、 **columnHeaders** パネルの最後の行がソート行になります。

継承元 **FlexGrid**
型 **number**

- stickyHeaders

ユーザーがドキュメントをスクロールしたときに、列ヘッダーを表示したままにするかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

- topLeftCells

左上のセル（列ヘッダーの左）を含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

treeIndent

異なるレベルの行グループをオフセットするインデントを取得または設定します。

継承元 **FlexGrid**
型 **number**

updatedLayoutNg

プログラムによるアクセスに使用されるWijmo **updatedLayout**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatedLayout** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatedViewNg

プログラムによるアクセスに使用されるWijmo **updatedView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatedView** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatingLayoutNg

プログラムによるアクセスに使用されるWijmo **updatingLayout**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatingLayout** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatingViewNg

プログラムによるアクセスに使用されるWijmo **updatingView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatingView** Wijmoイベント名を使用してください。

型 **EventEmitter**

validateEdits

検証に失敗した編集をユーザーがコミットしようとしたときに、グリッドを編集モードのままにするかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**getError** メソッドを呼び出して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

viewRange

現在表示されているセルの範囲を取得します。

継承元 **FlexGrid**
型 **CellRange**

virtualizationThreshold

仮想化を有効にするために必要な最小行数、最小列数、またはその両方を取得または設定します。

このプロパティはデフォルトでゼロに設定されます。つまり、仮想化は常に有効ということです。これにより、バインディングパフォーマンスとメモリー必要量が向上しますが、スクロール時のパフォーマンス低下は犠牲になります。

グリッドの行数が少ない場合（約50〜100）、このプロパティを150などのやや高い値に設定することで、スクロールのパフォーマンスを向上させることができます。これにより、仮想化が無効になり連結処理が遅くなりますが、認識されるスクロールのパフォーマンスが向上する可能性があります。たとえば、以下のコードは、データソースに150を超える項目がある場合、グリッドがセルを仮想化します。

```
// 150を超える項目がある場合はグリッドを仮想化します
theGrid.virtualizationThreshold = 150;
```

このプロパティを200より大きい値に設定することは推奨されません。ロード処理の時間が長くなり、グリッドはすべての行のセルを作成しているときに数秒間フリーズするため、ページ上の要素数が多いためブラウザが遅くなります。

行と列に別々の仮想化しきい値を設定する場合は、**virtualizationThreshold** プロパティを2つの数値を含む配列に設定できます。この場合、最初の数値は行のしきい値として使用され、2番目の数値は列のしきい値として使用されます。たとえば、次のコードでは、グリッドは行を仮想化しますが、列を仮想化しません。

```
// 行（しきい値0）は仮想化しますが、列は仮想化しません（しきい値10,000）
theGrid.virtualizationThreshold = [0, 10000];
```

継承元 **FlexGrid**
型 **any**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **className**s: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 `Control`
戻り値 **HTMLElement**

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

列をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **c**: **number**
  サイズ変更する列のインデックス。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 `FlexGrid`
戻り値 **void**

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

列の範囲をその内容がちょうど収まるようにサイズ変更します。

測定される行の範囲は常に、現在表示されているすべての行 + 現在表示されていない最大2,000行です。グリッドに大量のデータが含まれている場合には（たとえば、50,000行など）、すべての行を測定すると非常に時間がかかる可能性があるため、すべての行は測定されません。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstColumn**: **number** OPTIONAL
  サイズ変更する最初の列のインデックス（デフォルトは最初の列）。
- **lastColumn**: **number** OPTIONAL
  サイズ変更する最後の列のインデックス（デフォルトは最後の列）。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

行をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **r**: **number**
  サイズ変更する行のインデックス。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の列であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

行の範囲をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックス。
- **lastRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックスす。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の行であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

canEditCell

```
canEditCell(r: number, c: number): void
```

指定されたセルが編集可能かどうかを示す値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

継承元 **FlexGrid**
戻り値 **void**

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

すべてのグループ行を指定したレベルに折りたたみます。

パラメーター

- **level**: **number**
  表示する最大のグループレベル。

継承元 **FlexGrid**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

継承元 **FlexGrid**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

継承元 **FlexGrid**
戻り値 **boolean**

focus

```
focus(): void
```

グリッド全体をスクロールしてビューに入れることなくグリッドにフォーカスを設定するようにオーバーライドされます。

継承元 **FlexGrid**
戻り値 **void**

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セル要素の範囲（ビューポート座標単位）を取得します。

このメソッドは、**cells** パネル内のセル（スクロール可能なデータセル）の範囲を返します。その他のパネルにあるセルの範囲を取得するには、該当する**GridPanel** オブジェクトの**getCellBoundingRect** メソッドを使用してください。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

継承元 **FlexGrid**
戻り値 **Rect**

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

グリッドのスクロール可能領域内のセルに格納された値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

継承元 **FlexGrid**
戻り値 **any**

getClipString

```
getClipString(rng?: CellRange): string
```

CellRange の内容を、クリップボードへのコピーに適した文字列として取得します。

非表示の行および列はクリップボード文字列に含まれません。

パラメーター

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **string**

getColumn

```
getColumn(name: string): Column
```

名前または連結に基づいて列を取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **string**
  検索する名前または連結。

継承元 **FlexGrid**
戻り値 **Column**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

GridPanel 内のセルの結合範囲を示す**CellRange** を取得します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。

- **r**: **number**
  セルを含む行のインデックス。

- **c**: **number**
  セルを含む列のインデックス。

- **clip**: **boolean** OPTIONAL
  結合範囲をグリッドの現在のビュー範囲にクリップするかどうか。

継承元 **FlexGrid**
戻り値 **CellRange**

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

パラメーター

- **r**: **number**
  検査するセルの行インデックス。

- **c**: **number**
  検査するセルの列インデックス。

継承元 **FlexGrid**
戻り値 **SelectedState**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

指定されたポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

次に例を示します。

```
// ユーザーがグリッドをクリックしたときに、ポイントをヒットテストします
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

パラメーター

- **pt**: **any**
  調べる**Point** （ページ座標単位）、MouseEventオブジェクト、またはポイントのX座標。

- **y**: **any** OPTIONAL
  ポイントのY座標（ページ座標単位、最初のパラメーターが数値の場合）。

継承元 **FlexGrid**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

指定したCellRangeがこのグリッドの行および列コレクションに対して有効かどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックする範囲。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

autoSizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

autoSizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

autoSizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

autoSizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

beginningEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

cellEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

cellEditEnding イベントを発生させます。

パラメーター

- **e**: **CellEditEndingEventArgs**
  イベントデータを含む**CellEditEndingEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

copied イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

copying イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

deletedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

deletingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

draggedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

draggedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

draggingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

draggingColumnOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

draggingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

draggingRowOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

groupCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

groupCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onItemsSourceChanging

```
onItemsSourceChanging(e: CancelEventArgs): boolean
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

loadedRows イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

loadingRows イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

pasted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

pastedCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

pasting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

pastingCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

prepareCellForEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

resizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

resizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

resizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

resizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

rowAdded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

rowEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

rowEditEnding イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

rowEditStarted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

rowEditStarting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

scrollPositionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

selectionChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

sortedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

sortingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

updatedLayout イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

updatingLayout イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

updatingView イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

refresh

```
refresh(fullUpdate?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

継承元 **FlexGrid**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean**
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。
- **recycle**: **boolean** OPTIONAL
  既存の要素を再利用するかどうか。
- **state**: **boolean** OPTIONAL
  既存の要素を保持してその状態を更新するかどうか。

継承元 `FlexGrid`
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この`Control` が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 `Control`
戻り値 **number**

scrollIntoView

```
scrollIntoView(r: number, c: number, refresh?: boolean): boolean
```

指定したセルが画面に入るようにグリッドをスクロールします。

負の行と列のインデックスは無視されるので、以下を呼び出すと、

```
grid.scrollIntoView(200, -1);
```

グリッドは200行目を表示するように垂直にスクロールしますが、水平にスクロールしません。

パラメーター

- **r**: **number**
  画面に入るようにスクロールする行のインデックス
- **c**: **number**
  画面に入るようにスクロールする列のインデックス。
- **refresh**: **boolean** OPTIONAL
  スクロールした新しい位置をすぐ表示するためにグリッドを更新する必要があるかどうかを決定するオプションのパラメータ。

継承元 **FlexGrid**
戻り値 **boolean**

select

```
select(rng: any, show?: any): void
```

セル範囲を選択し、オプションでそのセル範囲が画面に入るようにスクロールします。

select メソッドは、**CellRange** 、と新しい選択範囲を画面に入るようにスクロールするかどうかを示すオプションのブール型パラメータを渡すことで呼び出すことができます。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(new CellRange(1, 1), true);

// 選択範囲（1,1）~（2,4）を指定し、画面に入るようにスクロールしません。
grid.select(new CellRange(1, 1, 2, 4), false);
```

select メソッドを呼び出してインデックスまたは選択する行と列を渡すこともできます。この場合、選択範囲が画面に入るようにスクロールします。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(1, 1);
```

パラメーター

- **rng**: **any**
  選択する範囲。
- **show**: **any** OPTIONAL
  新しい選択範囲が画面に入るようにスクロールするかどうか。

継承元 **FlexGrid**
戻り値 **void**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

グリッドのスクロール可能領域内のセルの値を設定します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **value**: **any**
  セルに格納する値。
- **coerce**: **boolean** OPTIONAL
  列のデータ型に合わせて値を自動的に変更するかどうか。
- **invalidate**: **boolean** OPTIONAL
  グリッドを無効にして変更を表示するかどうか。

継承元 **FlexGrid**
戻り値 **boolean**

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

文字列を行および列に解析し、その内容を特定の範囲に適用します。

非表示の行および列はスキップされます。

パラメーター

- **text**: **string**
  グリッドに解析する、タブと改行で区切られたテキスト。
- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **void**

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

指定されたセルの編集を開始します。

FlexGrid の編集は、Excelの編集によく似ています。 ［F2］キーを押すか、セルをダブルクリックすると、 グリッドが**完全編集** モードになります。 このモードでは、ユーザーが［Enter］、［Tab］、または［Esc］キーを押すか、マウスを使用して選択範囲を移動するまで、セルエディタはアクティブのままになります。完全編集モードでは、カーソルキーを押しても、グリッドの編集モードは終了しません。

テキストをセルに直接入力すると、グリッドは**クイック編集**モードになります。 このモードでは、ユーザーがEnter、Tab、Esc、またはいずれかの矢印キーを押すまで、セルエディタはアクティブなままになります。

通常、完全編集モードは既存の値を変更する際に使用します。 クイック編集モードは新しいデータをすばやく入力する際に使用します。

編集中に［F2］キーを押すことで、完全編集モードとクイック編集モードを切り替えることができます。

パラメーター

- **fullEdit**: **boolean** OPTIONAL
  ユーザーがカーソルキーを押したときも編集モードのままにするかどうか。 デフォルトはtrueです。
- **r**: **number** OPTIONAL
  編集する行のインデックス。 デフォルトは現在選択されている行です。
- **c**: **number** OPTIONAL
  編集する列のインデックス。 デフォルトは現在選択されている列です。
- **focus**: **boolean** OPTIONAL
  編集開始時に、エディタにフォーカスを与えるかどうか。 デフォルトはtrueです。

継承元 **FlexGrid**
戻り値 **boolean**

toggleDropDownList

```
toggleDropDownList(): void
```

現在選択されているセルに関連付けられたドロップダウンリストボックスを切り替えます。

このメソッドでは、セルが編集モードに入ったとき、またはユーザが特定のキーを押したときに、ドロップダウンリストを自動的に表示することができます。

たとえば、このコードでは、グリッドが編集モードに入るたびに、グリッドにドロップダウンリストが表示されます。

```
// グリッドが編集モードに入ると、ドロップダウンリストを表示する。
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

このコードでは、ユーザーがスペースバーを押した場合、グリッドが編集モードに入った後で、ドロップダウンリストが表示されます。

```
// ユーザーがスペースバーを押したときにドロップダウンリストを表示する。
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

継承元 **FlexGrid**
戻り値 **void**

イベント

autoSizedColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizedRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

beginningEdit

セルが編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnded

セル編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnding

セル編集が終了するときに発生します。

このイベントを使用して検証を実行し、無効な編集を防ぐことができます。 たとえば、次のコードは、ユーザーが文字「a」を含まない値を入力できないようにします。このコードは、編集が適用される前に、編集前と編集後の値を取得する方法を示しています。

```
function cellEditEnding (sender, e) {

  // 編集前と編集後の値を取得します
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // newValに「a」が含まれていない場合は、編集をキャンセルします
  e.cancel = newVal.indexOf('a') < 0;
}
```

cancel パラメータをtrueに設定すると、編集された値が無視されて、グリッドでセルの元の値が維持されます。

また、**stayInEditMode** パラメータをtrueに設定すると、グリッドの編集モードが維持され、編集をコミットする前にユーザーが無効な値を修正できます。

継承元 **FlexGrid**
引数 **CellEditEndingEventArgs**

copied

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーした後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

copying

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーするときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラでコピー操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletedRow

ユーザーが［Del］キーを押して行を削除した後に発生します （**allowDelete** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletingRow

ユーザーが［Delete］キーを押して選択されている行を削除するときに発生します（**allowDelete** プロパティを参照）。

イベントハンドラで行の削除をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedColumn

ユーザーが列のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedRow

ユーザーが行のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumn

ユーザーが列のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumnOver

ユーザーが列を別の位置にドラッグするときに発生します。

ハンドラは、このイベントをキャンセルして、ユーザーが特定の位置に列をドロップしないようにすることができます。次に例を示します。

```
// ドラッグされている列を記憶します
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// 'sales'列がインデックス0にドラッグされないようにします
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRow

ユーザーが行のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRowOver

ユーザーが行を別の位置にドラッグするときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

formatItem

セルを表す要素が作成されたときに発生します。

このイベントを使用してセルを表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、グループ行のセルから'wj-wrap'クラスを削除します。

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

継承元 **FlexGrid**
引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

groupCollapsedChanged

グループが展開または折りたたまれた後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

groupCollapsedChanging

グループが展開または折りたたまれる直前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

itemsSourceChanged

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

itemsSourceChanging

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

loadedRows

グリッド行がデータソースの項目に連結された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

loadingRows

グリッド行がデータソースの項目に連結される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

pasted

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastedCell

ユーザーがクリップボードの内容をセルに貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pasting

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた時に発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastingCell

ユーザーがクリップボードの内容をセルに貼り付けるときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

prepareCellForEdit

エディタセルが作成されたとき、それがアクティブになる前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

resizedColumn

ユーザーが列のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizedRow

ユーザーが行のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingColumn

列がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingRow

行がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowAdded

ユーザーが新規行テンプレートを編集して新しい項目を作成したときに発生します（**allowAddNew** プロパティを参照）。

イベントハンドラで新しい項目の内容をカスタマイズしたり、新しい項目の作成をキャンセルしたりできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnded

行編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnding

行編集が終了するとき、変更が確定またはキャンセルされる前に発生します。

このイベントを **rowEditStarted** イベントと組み合わせて使用して、ディープ連結編集を元に戻す操作を実装できます。次に例を示します。

```
// 編集の開始時にディープ連結値を保存します
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.')> -1) { // ディープ連結
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// 編集がキャンセルされたときにディープ連結値を復元します
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // ユーザーによって編集がキャンセルされました
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarted

行が編集モードに入った後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarting

行が編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

scrollPositionChanged

コントロールがスクロールされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

selectionChanged

選択が変更された後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

selectionChanging

選択が変更される前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortedColumn

ユーザーが列ヘッダをクリックしてソートを適用した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortingColumn

ユーザーが列ヘッダをクリックしてソートを適用する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

updatedLayout

グリッドで内部レイアウトが更新された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

updatedView

グリッドが現在のビューを構成する要素の作成/更新を完了したときに発生します。

グリッドのビューは以下のような操作に応じて更新されます。

- グリッドまたはそのデータソースの更新
- 行または列の追加、削除、変更
- グリッドのサイズ変更またはスクロール
- 選択の変更

継承元 **FlexGrid**
引数 **EventArgs**

updatingLayout

グリッドで内部レイアウトが更新される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

updatingView

現在のビューを構成する要素の作成/更新をグリッドが開始したときに発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

WjFlexGridCellTemplate クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.grid

FlexGrid セルテンプレート用のAngular 2ディレクティブ。

wjFlexGridCellTemplateディレクティブは、**FlexGrid** 内の特定のセルタイプに 使用されるテンプレートを定義します。これは、**<template>**要素に定義され、 **CellTemplateType** を指定する **cellType**属性が含まれる必要があります。テンプレートのセルタイプに応じて、 **wjFlexGridCellTemplate**ディレクティブを含む**<template>**要素は、 **WjFlexGrid** または **WjFlexGridColumn** ディレクティブのいずれかの子である必要があります。

列固有のセルテンプレートは**wj-flex-grid-column** コンポーネントに含まれる必要があり、列固有でないセル（行ヘッダーや左上のセルなど）は **wj-flex-grid**コンポーネントに含まれる必要があります。

wjFlexGridCellTemplateディレクティブを含む**<template>**要素には、 Angular 2内挿式および他のコンポーネント/ディレクティブを含む 任意のHTMLフラグメントを含めることができます。

HTMLフラグメント内の連結では、セルコンテキストオブジェクトを含む**cell**ローカルテンプレート変数を、 そのセルに関連する**Column**、**Row**、および**Row.dataItem**オブジェクトを参照する **col**、**row**、および**item**プロパティと共に使用できます。

Groupや**CellEdit**などのセルタイプには、書式設定されていないセル値を含む **value**プロパティが別途用意されています。たとえば、次の**FlexGrid** コントロールでは、 行ヘッダー用、Country列の通常のセル用、およびその列ヘッダーセル用にテンプレートが 使用されています。

```
import * as wjGrid from 'wijmo/wijmo.angular2.grid';

@Component({
    directives: [wjGrid.WjFlexGrid, wjGrid.WjFlexGridColumn, wjGrid.WjFlexGridCellTemplate],
    template: `
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'RowHeader'" let-cell="cell">
    {{cell.row.index}}
  </template>
  <template wjFlexGridCellTemplate [cellType]="'RowHeaderEdit'">
    ...
  </template>

  <wj-flex-grid-column [header]="'Country'" [binding]="'country'">
    <template wjFlexGridCellTemplate [cellType]="'ColumnHeader'" let-cell="cell">
      <img src="resources/globe.png" />
        {{cell.col.header}}
    </template>
    <template wjFlexGridCellTemplate [cellType]="'Cell'" let-cell="cell">
      <img src="resources/{{cell.item.country}}.png" />
      {{cell.item.country}}
    </template>
  </wj-flex-grid-column>
  <wj-flex-grid-column [header]="'Sales'" [binding]="'sales'"></wj-flex-grid-column>
</wj-flex-grid>
`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

特定のセルタイプテンプレートの詳細については、 **CellTemplateType** 列挙のドキュメントを参照してください。

wjFlexGridCellTemplateディレクティブは、次の属性をサポートします。

cellType	テンプレートの適用先のセルタイプを定義する**CellTemplateType**値。
cellOverflow	セルの**style.overflow**プロパティ値を定義します。

cellType属性には、次の列挙値を指定できます。

Cell

通常の（データ）セルテンプレートを定義します。**WjFlexGridColumn** コンポーネントの子である必要があります。 たとえば、次のセルテンプレートはCountry列のセルのフラグを示します。

```
<wj-flex-grid-column [header]="'Country'" [binding]="'country'">
  <template wjFlexGridCellTemplate [cellType]="'Cell'" let-cell="cell">
    <img src="resources/{{cell.item.country}}.png" />
    {{cell.item.country}}
  </template>
</wj-flex-grid-column>
```

階層**FlexGrid**（**childItemsPath**プロパティが指定されているグリッド）で **Group**テンプレートが提供されていない場合は、この**Column** のグループ行のヘッダー以外のセルにもこのテンプレートが使用されます。

CellEdit

編集モードのセルのテンプレートを定義します。**WjFlexGridColumn** コンポーネントの子である必要があります。 このセルタイプは、連結に利用できる**cell.value**プロパティを別途持ちます。また、 編集前の元のセル値と編集後の更新値を持ちます。 たとえば、Wijmo **InputNumber** コントロールを"Sales"列のエディタとして使用するテンプレートを次に示します。

```
<wj-flex-grid-column [header]="'Sales'" [binding]="'sales'">
  <template wjFlexGridCellTemplate [cellType]="'CellEdit'">
    <wj-input-number [(value)]="cell.value" [step]="1"></wj-input-number>
  </template>
</wj-flex-grid-column>
```

ColumnHeader

列ヘッダーセルのテンプレートを定義します。**WjFlexGridColumn** コンポーネントの子である必要があります。 たとえば、次のテンプレートは"Country"列のヘッダーに画像を追加します。

```
<wj-flex-grid-column [header]="'Country'" [binding]="'country'">
  <template wjFlexGridCellTemplate [cellType]="'ColumnHeader'" let-cell="cell">
    <img src="resources/globe.png" />
      {{cell.col.header}}
  </template>
</wj-flex-grid-column>
```

RowHeader

行ヘッダーセルのテンプレートを定義します。**WjFlexGrid** コンポーネントの子である必要があります。 たとえば、次のテンプレートは行ヘッダーに行インデックスを表示します。

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'RowHeader'" let-cell="cell">
    {{cell.row.index + 1}}
  </template>
</wj-flex-grid>
```

行ヘッダーセルが編集モードの行にある場合でも、このテンプレートが適用されます。 編集モードバージョンの行ヘッダーセルに別のコンテンツを提供するには、 **RowHeaderEdit**テンプレートを定義します。

RowHeaderEdit

編集モードの行ヘッダーセルのテンプレートを定義します。 **WjFlexGrid** コンポーネントの子である必要があります。たとえば、次のテンプレートは編集中の行のヘッダーにドットを表示します。

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'RowHeaderEdit'">
    ...
  </template>
</wj-flex-grid>
```

RowHeaderテンプレートが適用されるセルに標準的な編集モードインジケータを追加するには、 次の**RowHeaderEdit**テンプレートを使用します。

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'RowHeaderEdit'">
    {{&#x270e;}}
  </template>
</wj-flex-grid>
```

TopLeft

左上のセルのテンプレートを定義します。**WjFlexGrid** コンポーネントの子である必要があります。 たとえば、次のテンプレートはグリッドの左上のセルに右下三角グリフを表示します。

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'TopLeft'">
    <span class="wj-glyph-down-right"></span>
  </template>
</wj-flex-grid>
```

GroupHeader

GroupRow 内にあるグループヘッダーセルのテンプレートを定義します。 **WjFlexGridColumn** コンポーネントの子である必要があります。

cell.rowプロパティには、**GroupRow**クラスのインスタンスが含まれます。グループ化が **CollectionView** で行われる場合、**cell.item**プロパティは**CollectionViewGroup** オブジェクトを参照します。

たとえば、次のテンプレートは展開/折りたたみの切り替えにチェックボックス要素を使用します。

```
<wj-flex-grid-column [header]="'Country'" [binding]="'country'">
  <template wjFlexGridCellTemplate [cellType]="'GroupHeader'" let-cell="cell">
    <input type="checkbox" [(ngModel)]="cell.row.isCollapsed"/>
    {{cell.item.name}} ({{cell.item.items.length}} items)
  </template>
</wj-flex-grid-column>
```

Group

GroupRow 内にある通常のセル（グループヘッダーでない）のテンプレートを定義します。 **WjFlexGridColumn** コンポーネントの子である必要があります。このセルタイプは、連結に利用できる**cell.value**プロパティを別途持ちます。 列に**aggregate**プロパティが指定されている場合、その列には書式設定されていない集計値が含まれます。

たとえば、次のテンプレートは"Sales"列に集計値とグループ行セルの種類を表示します。

```
<wj-flex-grid-column [header]="'Sales'" [binding]="'sales'" [aggregate]="'Avg'">
  <template wjFlexGridCellTemplate [cellType]="'Group'" let-cell="cell">
    平均：{{cell.value | number:'1.0-0'}}
  </template>
</wj-flex-grid-column>
```

ColumnFooter

columnFooters パネル内にある通常のセルのテンプレートを定義します。 **WjFlexGridColumn** コンポーネントの子である必要があります。このセルタイプは、連結に利用できる、セル値を含む**cell.value**プロパティを別途持ちます。

たとえば、次のテンプレートは"Sales"列に集計値とグループ行セルの種類を表示します。

```
<wj-flex-grid-column [header]="'Sales'" [binding]="'sales'" [aggregate]="'Avg'">
  <template wjFlexGridCellTemplate [cellType]="'ColumnFooter'" let-cell="cell">
    Average: {{cell.value | number:'1.0-0'}}
  </template>
</wj-flex-grid-column>
```

BottomLeft

左下のセル（行ヘッダーと列フッターが交差する位置にあるセル）のテンプレートを定義します。 **WjFlexGrid** コンポーネントの子である必要があります。

たとえば、次のテンプレートはグリッドの左下のセルにシグマグリフを表示します

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'BottomLeft'">
    &#931;
  </template>
</wj-flex-grid>
```

NewCellTemplate 新しい行テンプレートのセルを定義します。**WjFlexGridColumn** コンポーネントの子である必要があります。 このタイプのセルに**cell.item**プロパティは定義されません。 たとえば、次のセルテンプレートは「新しい行」項目のDate列セルにプレースホルダを表示します。

```
<wj-flex-grid-column [header]="'Date'" [binding]="'date'">
  <template wjFlexGridCellTemplate [cellType]="'NewCellTemplate'">
    ここに日付を入力
  </template>
</wj-flex-grid-column>
```

プロパティ

- autoSizeRows
- cellOverflow
- forceFullEdit

プロパティ

- autoSizeRows

セルテンプレートがセルの内容に合わせてグリッドのデフォルトの行の高さを増加させるかどうかを示す値を取得または設定します。デフォルトはtrueです。

型 **boolean**

- cellOverflow

セルの**style.overflow**値を定義します。

型 **string**

- forceFullEdit

セル編集テンプレートでは、編集がどのように開始されたかにかかわらず、 セル編集が完全編集モードで強制的に開始するかどうかを示します。 完全編集モードでは、カーソルキーを押しても編集が終了しません。 デフォルトはtrueです。

型 **boolean**

WjFlexGridColumn クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid**
基本クラス **Column**
表示 ☐継承されたメンバー ☐イベント発生元

Column コントロールに対応するAngular 2コンポーネント。

wj-flex-grid-columnコンポーネントは、 **WjFlexGrid** コンポーネントに含める必要があります。

wj-flex-grid-columnコンポーネントを使用して、Angular 2アプリケーションに**Column**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexGridColumnコンポーネントは、**Column**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-flex-grid-columnコンポーネントには、 **WjFlexGridCellTemplate** 子ディレクティブを含めることができます。

コンストラクタ

- constructor

プロパティ

- aggregate
- align
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- asyncBindings
- binding
- collectionView
- cssClass
- currentSort
- dataMap
- dataType
- describedById
- dropDownCssClass
- format
- grid
- header
- index
- initialized
- inputType
- isContentHtml
- isInitialized
- isReadOnly
- isRequired
- isSelected
- isVisible
- mask
- maxLength
- maxWidth
- minWidth

- multiLine
- name
- pos
- quickAutoSize
- renderSize
- renderWidth
- showDropDown
- size
- sortMemberPath
- visible
- visibleIndex
- width
- wjProperty
- wordWrap

メソッド

- created
- getAlignment
- getIsRequired
- onPropertyChanged

コンストラクタ

constructor

```
constructor(options?: any): Column
```

Column クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  列の初期化オプション。

継承元 **Column**
戻り値 **Column**

プロパティ

- aggregate

列のグループヘッダ行に表示する **Aggregate** を取得または設定します。

継承元 **Column**
型 **Aggregate**

- align

列の項目の水平方向の配置を取得または設定します。

このプロパティのデフォルト値はnullで、列の**dataType** に基づいて配置が自動的に選択されます（数値の場合は右揃え、ブール値の場合は中央揃え、その他の型の場合は左揃え）。

デフォルトの配置をオーバーライドする場合は、 このプロパティを'left'、'right'、'center'のいずれかに設定します。

継承元 **Column**
型 **string**

- allowDragging

ユーザーがマウスで行または列を新しい位置に移動できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- allowMerging

行または列にあるセルを結合できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- allowResizing

ユーザーがマウスで行または列をサイズ変更できるかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- allowSorting

ユーザーがヘッダーをクリックして列をソートできるかどうかを示す値を取得または設定します。

継承元 **Column**
型 **boolean**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- binding

列がバインドされているプロパティの名前を取得または設定します。

継承元 **Column**
型 **string**

collectionView

この行または列にバインドされた**ICollectionView** を取得します。

継承元 **RowCol**
型 **ICollectionView**

cssClass

行または列のヘッダ以外のセルをレンダリングするときに使用するCSSクラス名を取得または設定します。

継承元 **RowCol**
型 **string**

currentSort

列に現在適用されているソート順序を表す文字列を取得します。 有効な値は、昇順の場合は'+'、降順の場合は'-'、列がソートされていない場合はnullです。

継承元 **Column**
型 **string**

dataMap

生の値から列の表示値への変換に使用される**DataMap** を取得または設定します。

dataMap が関連付けられた列には、値をすばやく編集するためのドロップダウンボタンが表示されます。ドロップダウンボタンが表示されないようにするには、列の**showDropDown** プロパティをfalseに設定します。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **Column**
型 **DataMap**

dataType

列に格納される値の型を取得または設定します。

グリッドを編集するとき、値は適切な型に型変換されます。

継承元 **Column**
型 **DataType**

describedById

列の説明を含む要素のIDを取得または設定します。

このIDは、列ヘッダ要素の **aria-describedby** 属性の値として使用されます。

継承元 **Column**
型 **string**

- dropDownCssClass

この列のドロップダウンに追加するCSSクラス名を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **Column**
型 **string**

- format

未加工の値を列の表示値に変換するために使用される書式文字列を取得または設定します （**Globalize** を参照）。

継承元 **Column**
型 **string**

- grid

行または列を所有する**FlexGrid** を取得します。

継承元 **RowCol**
型 **FlexGrid**

- header

列ヘッダに表示されるテキストを取得または設定します。

継承元 **Column**
型 **string**

- index

行または列の親コレクション内でのインデックスを取得します。

継承元 **RowCol**
型 **number**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- inputType

この列の値の編集に使用されるHTML入力要素の"type"属性を取得または設定します。

デフォルトでは、このプロパティは数値型の列では"tel"、その他のブール型以外の列型では"text"に設定されます。"tel"入力タイプを使用した場合、モバイルデバイスではマイナス記号や小数点を含む数値キーボードが表示されます。

デフォルトのままでは現在のカルチャ、デバイス、またはアプリケーションに関してうまく機能しない場合は、このプロパティを使用してデフォルト設定を変更します。その場合、値を"number"に設定するか、単に"text"にしてみてください。

継承元 `Column`
型 **string**

- isContentHtml

この行または列にあるセルがプレーンテキストではなくHTMLコンテンツを含むかどうかを示す値を取得または設定します。

継承元 `RowCol`
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

行または列のセルを編集できるかどうかを示す値を取得または設定します。

継承元 `RowCol`
型 **boolean**

- isRequired

列の値が必須かどうかを決定する値を取得または設定します。

デフォルトでは、このプロパティはnullに設定されます。その場合、値は必須ですが、 非マスク文字列の列に空の文字列を含めることができます。

trueに設定した場合、値は必須で、空の文字列は許可されません。

falseに設定した場合は、null値と空の文字列が許可されます。

継承元 `Column`
型 **boolean**

- isSelected

行または列が選択されているかどうかを示す値を取得または設定します。

継承元 `RowCol`
型 **boolean**

- isVisible

行または列が表示可能で、なおかつ折りたたまれていないかどうかを示す値を取得または設定します。

このプロパティは読み取り専用です。行または列の表示/非表示設定を変更するには、代わりに**visible** プロパティを使用してください。

継承元 **RowCol**
型 **boolean**

- mask

この列の値の編集時に使用するマスクを取得または設定します。

マスクの書式は**InputMask** コントロールで使用される書式と同じです。

指定する場合、このマスクは**format** プロパティの値と互換性がある必要があります。たとえば、'MM/dd/yyyy'と書式設定された 日付の入力にマスク'99/99/9999'を使用できます。

継承元 **Column**
型 **string**

- maxLength

セルに入力できる最大の項目数を取得または設定します。

このプロパティをnullに設定すると、任意の数の文字を入力できます。

継承元 **Column**
型 **number**

- maxWidth

列の最大幅を取得または設定します。

継承元 **Column**
型 **number**

- minWidth

列の最小幅を取得または設定します。

継承元 **Column**
型 **number**

- multiLine

この行または列にあるセルの内容が改行文字(\n).でラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

name

列の名前を取得または設定します。

列名を使用して、**getColumn** メソッドで列を取得できます。

継承元 **Column**
型 **string**

pos

行または列の位置を取得します。

継承元 **RowCol**
型 **number**

quickAutoSize

この列のサイズを自動変更するときにグリッドが精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、この列の自動リサイズ処理が無効になります。 これをtrueに設定すると、グリッドの**quickAutoSize** プロパティの値にしたがって、この機能が有効になります。 null（デフォルト値）に設定すると、プレーンテキストを表示するテンプレートを持たない列の機能が有効になります。

継承元 **Column**
型 **boolean**

renderSize

行または列のレンダリングサイズを取得します。 表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮したサイズが返されます。

継承元 **RowCol**
型 **number**

renderWidth

列のレンダリング幅を取得します。

列の表示/非表示設定、デフォルトサイズ、および最小/最大サイズを考慮した幅が返されます。

継承元 **Column**
型 **number**

showDropDown

グリッドで、この列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **Column**
型 **boolean**

- size

行または列のサイズを取得または設定します。 このプロパティをnullまたは負の値に設定すると、親コレクションのデフォルトサイズが使用されます。

継承元 **RowCol**
型 **number**

- sortMemberPath

この列をソートするときに使用するプロパティの名前を取得または設定します。

このプロパティは、**binding** プロパティによって指定されている値以外の値に基づいてソートを実行する場合に使用します。

このプロパティをnullに設定すると、**binding** プロパティの値を使用して列がソートされます。

継承元 **Column**
型 **string**

- visible

行または列が表示されているかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

- visibleIndex

非表示にした要素(**isVisible**)を無視し、親コレクション内の行または列のインデックスを取得します。

継承元 **RowCol**
型 **number**

- width

列の幅を取得または設定します。

列の幅は、正の数値（列幅のピクセル数を設定）、nullまたは負の数値（コレクションのデフォルトの列幅を使用）、または'{number}*'形式の文字列（スターサイズ指定）にすることができます。

スターサイズ指定オプションを使用すると、星の前の数字に比例して列の幅が決定されるXAMLスタイルの動的なサイズ設定が実行されます。たとえば、グリッドに3つの列があり、それぞれの幅が"100"、"*"、"3*"に設定されている場合、最初の列の幅は100ピクセルになり、2列目は残りスペースの1/4、3列目は残りスペースの3/4を占めます。

スターサイズ指定を使用すると、使用可能な幅いっぱいに自動的に広がる列を定義できます。たとえば、最後の列の幅を"*"に設定すると、最後の列がグリッドの幅いっぱいに拡張されて空スペースがなくなります。また、列の **minWidth** プロパティを設定して列の幅が狭くなりすぎないようにすることもできます。

継承元 **Column**
型 **any**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'columns'です。

型 **string**

wordWrap

この行または列のセルの内容を使用可能な列幅に収まれるようにラップするかどうかを示す値を取得または設定します。

継承元 **RowCol**
型 **boolean**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

getAlignment

```
getAlignment(): string
```

列の実際の配置を取得します。

align プロパティがnullでない場合はその値が返されます。それ以外の場合は、列の**dataType** に基づいて配置が選択されます。

継承元 **Column**
戻り値 **string**

getIsRequired

```
getIsRequired(): boolean
```

列が必須かどうかを判定する値を取得します。

isRequired プロパティがnullでない場合は、その値が返されます。 そうでない場合は、列の**dataType** に基づいて必須状態が判定されます。

デフォルトでは、文字列の列は、**dataMap** または **mask** 値が設定されていない限り 必須ではありません。他のすべてのデータ型は必須です。

継承元 **Column**
戻り値 **boolean**

onPropertyChanged

```
onPropertyChanged(): void
```

オーナーリストをダーティとしてマークし、オーナーグリッドを更新します。

継承元 **RowCol**
戻り値 **void**

CellTemplateType 列挙体

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid**

テンプレートの適用先のセルのタイプを定義します。この値は、**WjFlexGridCellTemplate** ディレクティブの **cellType**属性で指定されます。

メンバー

名前	値	説明
Cell	0	通常の（データ）セルを定義します。
CellEdit	1	編集モードのセルを定義します。
ColumnHeader	2	列ヘッダーセルを定義します。
RowHeader	3	行ヘッダーセルを定義します。
RowHeaderEdit	4	編集モードの行ヘッダーセルを定義します。
TopLeft	5	左上のセルを定義します。
GroupHeader	6	グループ行のグループヘッダーセルを定義します。
Group	7	グループ行の通常のセルを定義します。
NewCellTemplate	8	新しい行テンプレートのセルを定義します。
ColumnFooter	9	列フッターセルを定義します。
BottomLeft	10	左下のセル（行ヘッダーと列フッターが交差する位置にあるセル）を定義します。

wijmo/wijmo.angular2.grid.filter モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.grid.filter**

wijmo.grid.filterモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.grid.filterは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjFilter from 'wijmo/wijmo.angular2.grid.filter';
import * as wjGrid from 'wijmo/wijmo.angular2.grid';

@Component({
    directives: [wjGrid.WjFlexGrid, wjFilter.WjFlexGridFilter],
    template: `
      <wj-flex-grid [itemsSource]="data">
          <wj-flex-grid-filter [filterColumns]="['country', 'expenses']"></wj-flex-grid-filter>
      </wj-flex-grid>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjFlexGridFilter

WjFlexGridFilter クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid.filter**
基本クラス **FlexGridFilter**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGridFilter コントロールに対応するAngular 2コンポーネント。

wj-flex-grid-filterコンポーネントは、 **WjFlexGrid** コンポーネントに含める必要があります。

wj-flex-grid-filterコンポーネントを使用して、Angular 2アプリケーションに**FlexGridFilter**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexGridFilterコンポーネントは、**FlexGridFilter**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- activeEditor
- defaultFilterType
- filterAppliedNg
- filterChangedNg
- filterChangingNg
- filterColumns
- filterDefinition
- grid
- initialized
- isInitialized
- showFilterIcons
- showSortButtons
- wjProperty

メソッド

- apply
- clear
- closeEditor
- created
- editColumnFilter
- getColumnFilter
- onFilterApplied
- onFilterChanged
- onFilterChanging

イベント

- filterApplied
- filterChanged
- filterChanging

コンストラクタ

constructor

```
constructor(grid: FlexGrid, options?: any): FlexGridFilter
```

FlexGridFilter クラスの新しいインスタンスを初期化します。

パラメーター

- **grid**: **FlexGrid**
  フィルタ対象の**FlexGrid** 。
- **options**: **any** OPTIONAL
  FlexGridFilter の初期化オプション。

継承元 **FlexGridFilter**
戻り値 **FlexGridFilter**

プロパティ

activeEditor

アクティブな**ColumnFilterEditor** を取得します。

このプロパティを使用すると、**filterChanging** イベントを処理するときにフィルターエディターをカスタマイズできます。 フィルターが編集されていない場合はnullを返します。

継承元 **FlexGridFilter**
型 **ColumnFilterEditor**

defaultFilterType

使用するデフォルトフィルタタイプを取得または設定します。

この値は特定の列のフィルタでオーバーライドできます。 たとえば、以下のサンプルコードは、"ByValue"列を除くすべての列に対して条件に基づくフィルタを作成します。

```
var f = new wijmo.grid.filter.FlexGridFilter(flex);
f.defaultFilterType = wijmo.grid.filter.FilterType.Condition;
var col = flex.columns.getColumn('ByValue'),
    cf = f.getColumnFilter(col);
cf.filterType = wijmo.grid.filter.FilterType.Value;
```

The default value for this property is **FilterType.Both**.

継承元 **FlexGridFilter**
型 **FilterType**

filterAppliedNg

プログラムによるアクセスに使用されるWijmo **filterApplied**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **filterApplied** Wijmoイベント名を使用してください。

型 **EventEmitter**

filterChangedNg

プログラムによるアクセスに使用されるWijmo **filterChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **filterChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

filterChangingNg

プログラムによるアクセスに使用されるWijmo **filterChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **filterChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

filterColumns

フィルタを持つ列の名前またはバインディングを含む配列を取得または設定します。

このプロパティをnullまたは空の配列に設定すると、すべての列にフィルタが追加されます。

継承元 **FlexGridFilter**
型 **string[]**

filterDefinition

現在のフィルタ定義をJSON文字列として取得または設定します。

継承元 **FlexGridFilter**
型 **string**

grid

このフィルタを所有する**FlexGrid** への参照を取得します。

継承元 **FlexGridFilter**
型 **FlexGrid**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

showFilterIcons

FlexGridFilter がグリッドの列ヘッダにフィルタ編集ボタンを追加するかどうかを示す値を取得または設定します。

このプロパティをfalseに設定した場合は、ユーザーがフィルタを編集、クリア、および適用する手段を開発者が提供する必要があります。

The default value for this property is **true**.

継承元 **FlexGridFilter**
型 **boolean**

showSortButtons

フィルタエディタにソートボタンが表示されるかどうかを示す値を取得または設定します。

デフォルトでは、エディタにはExcelと同じようにソートボタンが表示されます。しかし、ユーザーはヘッダをクリックすることによって列をソートできるので、 フィルタエディタにソートボタンがあるのは望ましくない場合があります。

The default value for this property is **true**.

継承元 **FlexGridFilter**
型 **boolean**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は"です。

型 **string**

メソッド

apply

```
apply(): void
```

現在の列フィルタをグリッドに適用します。

継承元 **FlexGridFilter**
戻り値 **void**

clear

```
clear(): void
```

すべての列フィルタをクリアします。

継承元 **FlexGridFilter**
戻り値 **void**

closeEditor

```
closeEditor(): void
```

フィルタエディタを閉じます。

継承元 **FlexGridFilter**
戻り値 **void**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

editColumnFilter

```
editColumnFilter(col: any, ht?: HitTestInfo, refElem?: HTMLElement): void
```

指定したグリッド列のフィルタエディタを表示します。

パラメーター

- **col**: **any**
  編集するフィルタを含む**Column** 。
- **ht**: **HitTestInfo** OPTIONAL
  フィルタ表示をトリガしたセルの範囲を含む**HitTestInfo** オブジェクト。
- **refElem**: **HTMLElement** OPTIONAL
  An HTMLElement to use as a reference for positioning the editor.

継承元 **FlexGridFilter**
戻り値 **void**

getColumnFilter

```
getColumnFilter(col: any, create?: boolean): ColumnFilter
```

指定した列のフィルタを取得します。

パラメーター

- **col**: **any**
  フィルタの適用先の**Column** （または列名またはインデックス）。
- **create**: **boolean** OPTIONAL
  存在しない場合にフィルタを作成するかどうか。

継承元 **FlexGridFilter**
戻り値 **ColumnFilter**

onFilterApplied

```
onFilterApplied(e?: EventArgs): void
```

filterApplied イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGridFilter**
戻り値 **void**

onFilterChanged

```
onFilterChanged(e: CellRangeEventArgs): void
```

filterChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**

継承元 **FlexGridFilter**
戻り値 **void**

onFilterChanging

```
onFilterChanging(e: CellRangeEventArgs): boolean
```

filterChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGridFilter**
戻り値 **boolean**

イベント

filterApplied

フィルタが適用された後に発生します。

継承元 **FlexGridFilter**
引数 **EventArgs**

filterChanged

ユーザーが列フィルタを編集した後で発生します。

イベントパラメータを使用して、フィルタを所有する列を判定し、 変更が適用されたかキャンセルされたかを判定します。

継承元 **FlexGridFilter**
引数 **CellRangeEventArgs**

filterChanging

ユーザーが列フィルタを編集しようとしたときに発生します。

フィルタのデフォルトの設定をオーバーライドする場合は、 このイベントを使用して列フィルタをカスタマイズします。

たとえば、以下のコードは、フィルタ条件がnullの場合に、使用される演算子を 'contains'に設定します。

```
filter.filterChanging.addHandler(function (s, e) {
  var cf = filter.getColumnFilter(e.col);
  if (!cf.valueFilter.isActive && cf.conditionFilter.condition1.operator == null) {
    cf.filterType = wijmo.grid.filter.FilterType.Condition;
    cf.conditionFilter.condition1.operator = wijmo.grid.filter.Operator.CT;
  }
});
```

継承元 **FlexGridFilter**
引数 **CellRangeEventArgs**

wijmo/wijmo.angular2.grid.grouppanel モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid.grouppanel**

wijmo.grid.grouppanelモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.grid.grouppanelは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjPanel from 'wijmo/wijmo.angular2.grid.grouppanel';
import * as wjGrid from 'wijmo/wijmo.angular2.grid';

@Component({
    directives: [wjGrid.WjFlexGrid, wjPanel.WjGroupPanel],
    template: `
      <wj-group-panel
          [grid]="flex"
          [placeholder]="'ここに列をドラッグしてグループを作成します。'">
      </wj-group-panel>
      <wj-flex-grid #flex [itemsSource]="data">
      </wj-flex-grid>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjGroupPanel

WjGroupPanel クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid.grouppanel**
基本クラス **GroupPanel**
表示 ☐継承されたメンバー ☐イベント発生元

GroupPanel コントロールに対応するAngular 2コンポーネント。

wj-group-panelコンポーネントを使用して、Angular 2アプリケーションに**GroupPanel**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjGroupPanelコンポーネントは、**GroupPanel**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- filter
- gotFocusNg
- grid
- hideGroupedColumns
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- lostFocusNg
- maxGroups
- placeholder
- rightToLeft
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus

- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, options?): GroupPanel
```

GroupPanel クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **GroupPanel**
戻り値 **GroupPanel**

プロパティ

- STATIC controlTemplate

GroupPanel コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **GroupPanel**
型 **any**

- filter

Gets or sets the **FlexGridFilter** to use for filtering the grid data.

If you set this property to a valid filter, the group descriptors will display filter icons that can be used to see and edit the filer conditions associated with the groups.

継承元 **GroupPanel**
型 **FlexGridFilter**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- grid

この**GroupPanel** に接続している**FlexGrid** を取得または設定します。

グリッドをパネルに接続すると、 グリッドのデータソースで定義されているグループがパネルに表示されます。ユーザーはグリッド列をパネルにドラッグして新しいグループを作成できます。 また、パネル内でグループをドラッグしてグループ化の順序を変更したり、 パネルの項目を削除することによってグループを削除することもできます。

継承元 **GroupPanel**
型 **FlexGrid**

- hideGroupedColumns

グループ化列をオーナーグリッドで非表示にするかどうかを示す値を取得または設定します。

FlexGrid では行ヘッダにグループ化情報が表示されるので、 グループ化列を表示したままにすると情報が重複するため、 通常はグループ化列を非表示にすることを推奨します。

The default value for this property is **true**.

継承元 **GroupPanel**
型 **boolean**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

maxGroups

許可されるグループの最大数を取得または設定します。 Setting this property to -1 allows any number of groups to be created. Setting it to zero prevents any grouping.

The default value for this property is **6**.

継承元 **GroupPanel**
型 **number**

placeholder

グループがないときにコントロールに表示する文字列を取得または設定します。 The default value for this property is '' (empty string).

継承元 **GroupPanel**
型 **string**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(): void
```

パネルを更新して現在のグループを表示します。

継承元 **GroupPanel**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

wijmo/wijmo.angular2.grid.detail モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid.detail**

wijmo.grid.detailモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.grid.detailは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjDetail from 'wijmo/wijmo.angular2.grid.detail';
import * as wjGrid from 'wijmo/wijmo.angular2.grid';

@Component({
    directives: [wjGrid.WjFlexGrid, wjDetail.WjFlexGridDetail],
    template: `
      <wj-flex-grid [itemsSource]="data">
          <template wjFlexGridDetail>
              詳細行の内容をここに入れます...
          </template>
      </wj-flex-grid>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjFlexGridDetail

WjFlexGridDetail クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid.detail**
基本クラス **FlexGridDetailProvider**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid DetailRow テンプレートに対応するAngular 2ディレクティブ。

wj-flex-grid-detailディレクティブは、**wj-flex-grid**コンポーネントに含まれる **<template>**テンプレート要素で指定する必要があります。

wj-flex-grid-detailディレクティブは、詳細行の表示状態を保持する**FlexGridDetailProvider** クラスから派生され、詳細行のコンテンツは、 任意のHTMLフラグメントとしてディレクティブタグ内で定義されます。フラグメントには、 Angular 2連結、コンポーネント、およびディレクティブを含めることができます。 **row**および **item**テンプレート変数をAngular 2連結で使用して、 詳細行の親**Row** および**Row.dataItem**オブジェクトを参照できます。

コンストラクタ

- constructor

プロパティ

- createDetailCell
- detailVisibilityMode
- disposeDetailCell
- grid
- initialized
- isAnimated
- isInitialized
- keyActionEnter
- maxHeight
- rowHasDetail

メソッド

- created
- getDetailRow
- hideDetail
- isDetailAvailable
- isDetailVisible
- showDetail

コンストラクタ

constructor

```
constructor(grid: FlexGrid, options?: any): FlexGridDetailProvider
```

FlexGridDetailProvider クラスの新しいインスタンスを初期化します。

パラメーター

- **grid**: **FlexGrid**
  詳細行を受け取る**FlexGrid**。

- **options**: **any** OPTIONAL
  新しい**FlexGridDetailProvider** の初期化オプション。

継承元 **FlexGridDetailProvider**
戻り値 **FlexGridDetailProvider**

プロパティ

createDetailCell

詳細セルを作成するためのコールバック関数を取得または設定します。

このコールバック関数は、パラメータとして **Row** を受け取り、 行詳細を表すHTML要素を返します。 次に例を示します。

```
// 指定された行の詳細セルを作成します
dp.createDetailCell = function (row) {
  var cell = document.createElement('div');
  var detailGrid = new wijmo.grid.FlexGrid(cell, {
    itemsSource: getProducts(row.dataItem.CategoryID),
    headersVisibility: wijmo.grid.HeadersVisibility.Column
  });
  return cell;
};
```

継承元 **FlexGridDetailProvider**
型 **Function**

detailVisibilityMode

行詳細をいつ表示するかを決定する値を取得または設定します。 The default value for this property is **DetailVisibilityMode.ExpandSingle**.

継承元 **FlexGridDetailProvider**
型 **DetailVisibilityMode**

disposeDetailCell

詳細セルを破棄するためのコールバック関数を取得または設定します。

このコールバック関数は、パラメータとして **Row** を受け取り、 詳細セルに関連付けられているすべてのリソースを破棄します。

この関数はオプションです。この関数は、自動的に ガベージコレクトされないリソースを **createDetailCell** 関数で 割り当てている場合に使用します。

継承元 **FlexGridDetailProvider**
型 **Function**

grid

このFlexGridDetailProvider を所有するFlexGrid。

継承元 FlexGridDetailProvider
型 FlexGrid

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 EventEmitter

isAnimated

行詳細を表示するときにアニメーションを使用するかどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 FlexGridDetailProvider
型 boolean

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 boolean

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティのデフォルト設定は**None** です。これにより、グリッドはキーを処理できます。 そして**ToggleDetail** により、Enterキーを操作して詳細行の表示を切り替えます。

継承元 FlexGridDetailProvider
型 KeyAction

maxHeight

詳細行の最大高さをピクセル単位で取得または設定します。 The default value for this property is **null**, which means there's no upper limit to the detail row height.

継承元 FlexGridDetailProvider
型 number

rowHasDetail

行に詳細があるかどうかを判定するためのコールバック関数を取得または設定します。

このコールバック関数は、パラメータとして **Row** を受け取り、 行に詳細があるかどうかを示すブール値を返します。次に例を示します。

```
// 奇数のCategoryIDを持つ項目から詳細を削除します
dp.rowHasDetail = function (row) {
  return row.dataItem.CategoryID % 2 == 0;
};
```

このプロパティをnullに設定すると、すべての行に詳細が含まれます。

継承元 **FlexGridDetailProvider**
型 **Function**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

getDetailRow

```
getDetailRow(row: any): DetailRow
```

指定したグリッド行に関連付けられた詳細行を取得します。

パラメーター

- **row**: **any**
  調査する行または行のインデックス。

継承元 **FlexGridDetailProvider**
戻り値 **DetailRow**

hideDetail

```
hideDetail(row?: any): void
```

指定された行の詳細行を非表示にします。

パラメーター

- **row**: **any** OPTIONAL
  詳細が非表示になっている行または行のインデックス。このパラメータはオプションです。 指定されない場合は、すべての詳細行が非表示になります。

継承元 **FlexGridDetailProvider**
戻り値 **void**

isDetailAvailable

```
isDetailAvailable(row: any): boolean
```

行に表示する詳細があるかどうかを決定する値を取得します。

パラメーター

- **row**: **any**
  調査する行または行のインデックス。

継承元 **FlexGridDetailProvider**
戻り値 **boolean**

isDetailVisible

```
isDetailVisible(row: any): boolean
```

行の詳細を表示するかどうかを決定する値を取得します。

パラメーター

- **row**: **any**
  調査する行または行のインデックス。

継承元 **FlexGridDetailProvider**
戻り値 **boolean**

showDetail

```
showDetail(row: any, hideOthers?: boolean): void
```

指定された行の詳細行を表示します。

パラメーター

- **row**: **any**
  詳細が表示されている行または行のインデックス。

- **hideOthers**: **boolean** OPTIONAL
  他のすべての行の詳細を非表示にするかどうか。

継承元 **FlexGridDetailProvider**
戻り値 **void**

wijmo/wijmo.angular2.grid.multirow モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid.multirow**

wijmo.grid.multirowモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.grid.multirowは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjMultiRow from 'wijmo/wijmo.angular2.grid.multirow';

@Component({
    directives: [wjMultiRow.WjMultiRow],
    template: `<wj-multi-row></wj-multi-row>`,
    selector: 'my-cmp',
})
export class MyCmp {
}
```

クラス

- WjMultiRow

WjMultiRow クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid.multirow**
基本クラス **MultiRow**
表示 ☐継承されたメンバー ☐イベント発生元

MultiRow コ ントロールに対応するAngular 2コンポーネント。

wj-multi-rowコンポーネントを使用して、Angular 2アプリケーションに**MultiRow**コントロールを追加します。 Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjMultiRowコンポーネントは、**MultiRow**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoScroll
- autoSearch
- autoSizedColumnNg
- autoSizedRowNg
- autoSizeMode
- autoSizingColumnNg
- autoSizingRowNg
- beginningEditNg
- bottomLeftCells
- cellEditEndedNg
- cellEditEndingNg
- cellFactory
- cells
- centerHeadersVertically
- childItemsPath
- clientSize
- cloneFrozenCells
- collapsedHeaders
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate

- copiedNg
- copyingNg
- deferResizing
- deletedRowNg
- deletingRowNg
- draggedColumnNg
- draggedRowNg
- draggingColumnNg
- draggingColumnOverNg
- draggingRowNg
- draggingRowOverNg
- editableCollectionView
- editRange
- formatItemNg
- frozenColumns
- frozenRows
- gotFocusNg
- groupCollapsedChangedNg
- groupCollapsedChangingNg
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- initialized
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceChangedNg
- itemValidator
- keyActionEnter
- keyActionTab
- layoutDefinition
- loadedRowsNg
- loadingRowsNg
- lostFocusNg
- mergeManager
- newRowAtTop
- pastedCellNg
- pastedNg
- pastingCellNg
- pastingNg
- prepareCellForEditNg
- preserveOutlineState
- preserveSelectedState
- quickAutoSize

- quickAutoSize
- resizedColumnNg
- resizedRowNg
- resizingColumnNg
- resizingRowNg
- rightToLeft
- rowAddedNg
- rowEditEndedNg
- rowEditEndingNg
- rowEditStartedNg
- rowEditStartingNg
- rowHeaderPath
- rowHeaders
- rows
- rowsPerItem
- scrollPosition
- scrollPositionChangedNg
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionChangedNg
- selectionChangingNg
- selectionMode
- showAlternatingRows
- showDropDown
- showErrors
- showGroups
- showHeaderCollapseButton
- showMarquee
- showSelectedHeaders
- showSort
- sortedColumnNg
- sortingColumnNg
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- updatedLayoutNg
- updatedViewNg
- updatingLayoutNg
- updatingViewNg
- validateEdits
- viewRange
- virtualizationThreshold
- wjModelProperty

メソッド

- addEventListener

- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseGroupsToLevel
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getBindingColumn
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCollapsedHeadersChanged
- onCollapsedHeadersChanging
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver

- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onItemsSourceChanging
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onRefreshed
- onRefreshing
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- startEditing
- toggleDropDownList

イベント

- autoSizedColumn
- autoSizedRow
- autoSizingColumn

- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- collapsedHeadersChanged
- collapsedHeadersChanging
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- itemsSourceChanging
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- refreshed
- refreshing
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged
- selectionChanging
- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout

updatingView

コンストラクタ

constructor

```
constructor(element: any, options?): MultiRow
```

MultiRow クラスの新しいインスタンスを初期化します。

通常、**options**パラメータには**layoutDefinition** プロパティの 値が含まれます。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **MultiRow**
戻り値 **MultiRow**

プロパティ

activeEditor

現在アクティブになっているセルエディタを表す**HTMLInputElement**を取得します。

継承元 **FlexGrid**
型 **HTMLInputElement**

allowAddNew

ユーザーがソースコレクションに項目を追加できるようにグリッドに新規行テンプレートを表示するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、新規行テンプレートは表示されません。

継承元 **FlexGrid**
型 **boolean**

allowDelete

ユーザーが［Delete］キーを押したときにグリッドの選択されている行を削除するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、選択されている行は削除されません。

継承元 **FlexGrid**
型 **boolean**

allowDragging

ユーザーがマウスを使用して行、列、またはその両方をドラッグできるかどうかを決定する値を取得または設定します。

autoScroll プロパティがtrueに設定されている場合、ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされます。

グリッドでは、列のドラッグがデフォルトで有効になっています。

グリッドが連結モードでは、行をドラッグするには特別な 考慮が必要になります。

グリッドが連結モードでは、行をドラッグするとデータソースとの同期が失われます（たとえば行4は6行として参照されることがあります）。 これを回避するには、**draggedRow** イベントを処理し、データを新しい行の位置と同期させる必要があります。

また、**allowSorting** プロパティをfalseに設定する必要があります。そうしないと、行の順序がデータによって決定され、行をドラッグするのは無意味になります。

次のフィドルでは、連結グリッドでの行のドラッグを実行しています。 **Bound Row Dragging (http://jsfiddle.net/Wijmo5/kyg0qsda/)**.

継承元 **FlexGrid**
型 **AllowDragging**

allowMerging

グリッドのどの部分のセル結合を許可するかを取得または設定します。

継承元 **FlexGrid**
型 **AllowMerging**

allowResizing

ユーザーがマウスを使用して行、列、またはその両方をサイズ変更できるかどうかを決定する値を取得または設定します。

サイズ変更が可能な場合は、列ヘッダーセルの右端をドラッグして列を、行ヘッダーセルの下端をドラッグして行をサイズ変更できます。

ヘッダーセルの端をダブルクリックして、行および列をコンテンツに合わせて自動的にサイズ変更することもできます。 自動サイズ変更の動作は、**autoSizeMode** プロパティを使用してカスタマイズできます。

継承元 **FlexGrid**
型 **AllowResizing**

allowSorting

ユーザーが列ヘッダーセルをクリックして列をソートできるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

autoClipboard

グリッドがクリップボードショートカットを処理するかどうかを決定する値を取得または設定します。

次のクリップボードショートカットがあります。

［Ctrl］+［C］、［Ctrl］+［Ins］	グリッドの選択範囲をクリップボードにコピーします。
［Ctrl］+［V］、［Shift］+［Ins］	クリップボードのテキストをグリッドの選択範囲に貼り付けます。

クリップボード操作は、表示されている行および列だけが対象となります。

読み取り専用のセルは、貼り付け操作の影響を受けません。

継承元 **FlexGrid**
型 **boolean**

autoGenerateColumns

グリッドが**itemsSource** に基づいて列を自動的に生成するかどうかを 決定する値を取得または設定します。

列の生成は、少なくとも1項目を含む**itemsSource** プロパティに依存します。このデータ項目が検査され、列が作成されて、プリミティブ値（数値、文字列、Boolean、またはDate）を含むプロパティに連結されます。

プロパティをnullに設定すると、適切なタイプを推測する方法がないため、列は生成されません。このような場合は、**autoGenerateColumns** プロパティをfalseに設定し、明示的に列を作成する必要があります。次に例を示します。

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // データ項目にnull値が含まれる場合があります
  columns: [                  // そのため、列を明示的に定義します
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

継承元 **FlexGrid**
型 **boolean**

autoScroll

ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされるかどうかを決定する値を取得または設定します。

行と列のドラッグは、**allowDragging** プロパティによって制御されます。

このプロパティはデフォルトでtrueに設定されます。

継承元 **FlexGrid**
型 **boolean**

autoSearch

ユーザーが読み取り専用セルに入力するに伴ってグリッドでセルを検索するかどうかを決定する値を取得または設定します。

検索が現在選択されている列(編集不可能な場合)で行われます。検索は現在選択されている行から始まり、大文字と小文字を区別されません。

継承元 **FlexGrid**
型 **boolean**

autoSizedColumnNg

プログラムによるアクセスに使用されるWijmo **autoSizedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

autoSizedRowNg

プログラムによるアクセスに使用されるWijmo **autoSizedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

autoSizeMode

行または列のサイズを自動設定するときにどのセルを考慮に入れるかを取得または設定します。

このプロパティは、ユーザーが列ヘッダの端をダブルクリックしたときの動作を制御します。

デフォルトでは、その列のヘッダセルとデータセルの内容に基づいて列の幅が自動的に設定されます。このプロパティを使用すると、ヘッダのみまたはデータのみに基づいて列の幅を設定するように変更できます。

継承元 **FlexGrid**
型 **AutoSizeMode**

autoSizingColumnNg

プログラムによるアクセスに使用されるWijmo **autoSizingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

autoSizingRowNg

プログラムによるアクセスに使用されるWijmo **autoSizingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

beginningEditNg

プログラムによるアクセスに使用されるWijmo **beginningEdit**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **beginningEdit** Wijmoイベント名を使用してください。

型 **EventEmitter**

bottomLeftCells

左下のセルを含む**GridPanel** を取得します。

bottomLeftCells パネルは、行ヘッダの下、**columnFooters** パネルの左に表示されます。

継承元 **FlexGrid**
型 **GridPanel**

cellEditEndedNg

プログラムによるアクセスに使用されるWijmo **cellEditEnded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **cellEditEnded** Wijmoイベント名を使用してください。

型 **EventEmitter**

cellEditEndingNg

プログラムによるアクセスに使用されるWijmo **cellEditEnding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **cellEditEnding** Wijmoイベント名を使用してください。

型 **EventEmitter**

cellFactory

このグリッドのセルを作成および更新する**CellFactory** を取得または設定します。

継承元 **FlexGrid**
型 **CellFactory**

cells

データセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

centerHeadersVertically

複数の行にまたがるセルのコンテンツを垂直方向に中央揃えにするかどうかを 判定する値を取得または設定します。

The default value for this property is **true**.

継承元 **MultiRow**
型 **boolean**

childItemsPath

階層グリッドで子の行の生成に使用される 1 つ以上のプロパティの名前を取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `childItemsPath = 'items'` ）。

項目の異なるレベルに異なる名前を持つ子項目がある場合は、このプロパティに、各レベルの子項目を含むプロパティの名前から成る配列を設定します。（たとえば、 `childItemsPath = ['checks','earnings'];` ）。

サンプル

Show me (https://jsfiddle.net/Wijmo5/kk9j93bL)

継承元 **FlexGrid**
型 **any**

clientSize

コントロールのクライアントサイズを取得します（コントロールのサイズからヘッダーとスクロールバーを引いた値）。

継承元 **FlexGrid**
型 **Size**

cloneFrozenCells

スクロール中に知覚されるパフォーマンスを向上させるには、FlexGridがフリーズされたセルを複製し、別の要素に表示するかどうかを決定する値を取得または設定します。

このプロパティのデフォルト値はnullであり、ブラウザーによって一番適当な設定が選択されます。

継承元 **FlexGrid**
型 **boolean**

collapsedHeaders

列ヘッダーを折りたたみ、単一の行としてグループヘッダーを 表示するかどうかを判定する値を取得または設定します。

collapsedHeadersプロパティをtrueに設定した場合は、 各グループの **header**プロパティを設定して、ヘッダーが空に ならないようにしてください。

`collapsedHeaders` プロパティを**null**に設定すると、 グリッドにすべてのヘッダー情報（グループおよび列）が表示されます。 この場合、最初の行にはグループヘッダーが、残りの行には個々の 列ヘッダーが表示されます。

The default value for this property is **false**.

継承元 **MultiRow**
型 **boolean**

collectionView

グリッドデータを含む**ICollectionView** を取得します。

継承元 **FlexGrid**
型 **ICollectionView**

columnFooters

列フッターセルを保持する**GridPanel** を取得します。

columnFooters パネルは、グリッドセルの下、**bottomLeftCells** パネルの右に表示されます。これを使用して、グリッドデータの下にサマリー情報を表示できます。

以下の例では、**columnFooters** パネルに 1行を追加して、**aggregate** プロパティが設定された列に サマリーデータを表示する方法を示します。

```
function addFooterRow(flex) {

  // 集計を表示するGroupRowを作成します
  var row = new wijmo.grid.GroupRow();

  // 列フッターパネルに行を追加します
  flex.columnFooters.rows.push(row);

  // ヘッダーにシグマを表示します
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

継承元 **FlexGrid**
型 **GridPanel**

columnHeaders

列ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

columnLayout

現在の列レイアウトを定義するJSON文字列を取得または設定します。

列レイアウト文字列は、列とそのプロパティを含む配列を表します。これを使用して、ユーザーが定義した列レイアウトをセッションを越えて保持できます。また、ユーザーが列レイアウトを変更できるアプリケーションで元に戻す/やり直し機能を実装することもできます。

データマップはシリアル化できないため、列レイアウト文字列に**dataMap**　プロパティは含まれていません。

継承元 **FlexGrid**
型 **string**

columns

グリッドの列コレクションを取得します。

継承元 **FlexGrid**
型 **ColumnCollection**

controlRect

コントロールの外接矩形（ページ座標単位）を取得します。

継承元 **FlexGrid**
型 **Rect**

STATIC controlTemplate

FlexGrid コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **FlexGrid**
型 **any**

copiedNg

プログラムによるアクセスに使用されるWijmo **copied**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **copied** Wijmoイベント名を使用してください。

型 **EventEmitter**

copyingNg

プログラムによるアクセスに使用されるWijmo **copying**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **copying** Wijmoイベント名を使用してください。

型 **EventEmitter**

deferResizing

ユーザーがマウスボタンを放すまで、行および列のサイズ変更を遅らせるかどうかを決定する値を取得または設定します。

デフォルトでは、**deferResizing** はfalseに設定されており、ユーザーがマウスをドラッグするに伴って行および列がサイズ変更されます。このプロパティをtrueに設定すると、グリッドにサイズ変更マーカーが表示され、ユーザーがマウスボタンを放したときに初めて行または列のサイズが変更されます。

継承元 **FlexGrid**
型 **boolean**

deletedRowNg

プログラムによるアクセスに使用されるWijmo **deletedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **deletedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

deletingRowNg

プログラムによるアクセスに使用されるWijmo **deletingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **deletingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggedColumnNg

プログラムによるアクセスに使用されるWijmo **draggedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggedRowNg

プログラムによるアクセスに使用されるWijmo **draggedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingColumnNg

プログラムによるアクセスに使用されるWijmo **draggingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingColumnOverNg

プログラムによるアクセスに使用されるWijmo **draggingColumnOver**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingColumnOver** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingRowNg

プログラムによるアクセスに使用されるWijmo **draggingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingRowOverNg

プログラムによるアクセスに使用されるWijmo **draggingRowOver**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingRowOver** Wijmoイベント名を使用してください。

型 **EventEmitter**

editableCollectionView

グリッドデータを含む**IEditableCollectionView** を取得します。

継承元 **FlexGrid**
型 **IEditableCollectionView**

editRange

現在編集中のセルを識別する**CellRange** を取得します。

継承元 **FlexGrid**
型 **CellRange**

formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

frozenColumns

固定された列の数を取得または設定します。

固定された列は水平方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

frozenRows

静止行の数を取得または設定します。

固定された行は垂直方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupCollapsedChangedNg

プログラムによるアクセスに使用されるWijmo **groupCollapsedChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **groupCollapsedChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupCollapsedChangingNg

プログラムによるアクセスに使用されるWijmo **groupCollapsedChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **groupCollapsedChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupHeaderFormat

グループヘッダーコンテンツを作成するために使用される書式文字列を取得または設定します。

この文字列は、任意のテキストと次の置換文字列を含むことができます。

- **{name}**：グループ化されるプロパティの名前。
- **{value}**：グループ化されるプロパティの値。
- **{level}**：グループレベル。
- **{count}**：このグループ内にある項目の合計数。

列がグループ化プロパティに連結されている場合は、列ヘッダーを使用して パラメータを置換し、列の書式とデータマップを使用して `{value}` パラメータを計算します。 列がない場合は、代わりにグループ情報が使用されます。

グループプロパティに連結された非表示の列を追加して、グループヘッダーセルの書式設定をカスタマイズすることもできます。

このプロパティのデフォルト値は
`'{name}: <b>{value}</b>({count:n0} items)'` です。 これは、 `'Country: UK (12 items)'` や
`'Country: Japan (8 items)'` のようなグループヘッダーを作成します。

継承元 **FlexGrid**
型 **string**

headersVisibility

行ヘッダおよび列ヘッダが表示されるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

imeEnabled

編集モードでないときに、グリッドがIME（Input Method Editor）をサポートするかどうかを決定する値を取得または設定します。

このプロパティは、日本語、中国語、韓国語など、IMEサポートを必要とする言語のサイト/アプリケーションにのみ関係します。

継承元 **FlexGrid**
型 **boolean**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用してセル値を変更できるかどうかを判定する値を取得または設定します。

継承元 `FlexGrid`
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

itemFormatter

このグリッドのセルのカスタマイズに使用されるフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意のセルに任意の内容を追加できます。そのため、グリッドセルの外観と動作を非常に柔軟に制御することが可能です。

この関数は、セルを含む**GridPanel**、セルの行インデックスと列インデックス、セルを表すHTML要素の4つのパラメーターをとります。通常は、関数内でセル要素の**innerHTML**　プロパティを変更するようにします。

例:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == wijmo.grid.CellType.Cell) {

    // セルにスパークラインを描画します。
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

FlexGridはセルをリサイクルするため、**itemFormatter** によってセルのスタイル属性を変更する場合は、 セルの適切でないスタイル属性を事前にリセットする必要があります。例:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // カスタマイズする属性をリセットします。
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // このセルのcolorおよびbackgroundColor属性をカスタマイズします。
  ...
}
```

複数のクライアントがグリッドのレンダリングをカスタマイズできるようにする場合は（たとえば、ディレクティブや再利用可能なライブラリを作成するときなど）、代わりに**formatItem** イベントを使用することを検討してください。このイベントを使用すると、複数のクライアントがそれぞれ独自のハンドラをアタッチできます。

継承元　　**FlexGrid**
型　　**Function**

itemsSource

グリッドに表示される項目を含む配列または**ICollectionView** を取得または設定します。

継承元　　**FlexGrid**
型　　**any**

itemsSourceChangedNg

プログラムによるアクセスに使用されるWijmo **itemsSourceChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemsSourceChanged** Wijmoイベント名を使用してください。

型　　**EventEmitter**

itemValidator

セルに有効なデータが含まれているかどうかを決定するバリデータ関数を取得または設定します。

指定された場合、バリデータ関数はセルの行インデックスと列インデックスを含む2つのパラメータをとり、エラーの説明を含む文字列を返す必要があります。

このプロパティは、バインドされていないグリッドを処理する場合に特に便利です。バインドされたグリッドは、代わりに**getError** プロパティを使用して検証できます。

この例では、セルのすぐ上のセルと同じデータがセルに含まれないようにする方法を示します。

```
// 上のセルは、現在のセルと同じ値が含まれていないことを確認します
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'これは重複した値です...'
    }
  }
  return null; // エラーなし
}
```

継承元 **FlexGrid**
型 **Function**

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は**MoveDown** となり、コントロールがカーソルを次の行に移動します。 この動作は標準的なExcel式の動作です。

継承元 **FlexGrid**
型 **KeyAction**

keyActionTab

［Tab］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は、**None** となります。これにより、Tabキーが押されたときにブラウザ上で次または前のコントロールを選択できます。これは、ページのアクセシビリティを向上させるために推奨される設定になります。

以前のバージョンでは、デフォルトは**Cycle** に設定されていました。これにより、コントロールがカーソルを、グリッドを横切って下部に移動しました。これは標準的なExcelの動作ですが、アクセシビリティには適していません。

また、**CycleOut** の設定が存在します。これにより、カーソルがセルを通じて移動（**Cycle** ）してから、最後または最初のセルが選択されたときにページの次/前のコントロールに移動します 。

継承元 **FlexGrid**
型 **KeyAction**

layoutDefinition

各データ項目の表示に使用される行のレイアウトを定義する配列を取得または設定します。

この配列には、次のプロパティを持つセルグループオブジェクトのリストが含まれます。

- **header**: グループヘッダー（ヘッダーが折りたたまれたときに表示される）
- **colspan**: グループがまたがるグリッド列の数
- **cells**: セルオブジェクトの配列。これらは、**colspan**プロパティで**Column** を拡張します。

`layoutDefinition` プロパティを設定すると、グリッドは各グループ内のセルを 次のようにスキャンします。

1. グリッドは、グループ自体のcolspanとグループ内の最も広いセルのスパンの 大きい方をグループのcolspanとします。
2. セルがグループ内の現在の行に収まる場合、そのセルは現在の行に追加されます。
3. 収まらない場合は、新しい行に追加されます。

すべてのグループが準備できたら、すべてのグループの最大のrowspanを1レコードの行数とし、 必要に応じて各グループに行を追加して高さを合わせます。

このスキームは単純で柔軟です。次に例を示します。

```
{ header: 'Group 1', cells: [{ binding: 'c1' }, { bnding: 'c2'}, { binding: 'c3' }]}
```

このグループのcolspanは1なので、各列に1つのセルが使用されます。結果は次のとおりです。

```
| C1 |
| C2 |
| C3 |
```

2列を含むグループを作成するには、グループの **colspan**プロパティを次のように設定します。

```
{ header: 'Group 1', colspan: 2, cells:[{ binding: 'c1' }, { binding: 'c2'}, { binding: 'c3' }]}
```

セルは次のように折り返されます。

```
| C1  | C2 |
| C3       |
```

最後のセルは2列にまたがります（グループ全体に拡大）。

グループではなく個々のセルにcolspanを指定することもできます。次に例を示します。

```
{ header: 'Group 1', cells: [{binding: 'c1', colspan: 2 }, { bnding: 'c2'}, { binding: 'c3' }]}
```

最初のセルのcolspanが2なので、結果は次のようになります。

```
| C1       |
| C2 |  C3 |
```

セルは**`Column`** クラスを拡張しているため、どのセルにも通常の **`Column`** プロパティのすべてを追加できます。 次に例を示します。

```
{ header: 'Group 1', cells: [
   { binding: 'c1', colspan: 2 },
   { bnding: 'c2'},
   { binding: 'c3', format: 'n0', required: false, etc... }}
]}
```

継承元 **`MultiRow`**
型 **any**[]

loadedRowsNg

プログラムによるアクセスに使用されるWijmo **loadedRows**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadedRows** Wijmoイベント名を使用してください。

型 **EventEmitter**

loadingRowsNg

プログラムによるアクセスに使用されるWijmo **loadingRows**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadingRows** Wijmoイベント名を使用してください。

型 **EventEmitter**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

mergeManager

セルの結合方法を決定する**MergeManager** オブジェクトを取得または設定します。

継承元 **FlexGrid**
型 **MergeManager**

newRowAtTop

新しい行テンプレートをグリッドの上部に配置するか、下部に配置するかを示す値を取得または設定します。

newRowAtTop プロパティをtrueに設定した場合、新しい行テンプレートを常に表示したままにする場合は、**frozenRows** プロパティを 1に設定します。これにより、新しい行テンプレートは上部に固定され、ビューからスクロールオフされなくなります。

allowAddNew プロパティがtrueに設定されている場合、および**itemsSource** オブジェクトが新しい項目の追加をサポートしている場合にのみ、新しい行テンプレートが表示されます。

継承元 **FlexGrid**
型 **boolean**

pastedCellNg

プログラムによるアクセスに使用されるWijmo **pastedCell**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pastedCell** Wijmoイベント名を使用してください。

型 **EventEmitter**

- pastedNg

プログラムによるアクセスに使用されるWijmo **pasted**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pasted** Wijmoイベント名を使用してください。

型 **EventEmitter**

- pastingCellNg

プログラムによるアクセスに使用されるWijmo **pastingCell**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pastingCell** Wijmoイベント名を使用してください。

型 **EventEmitter**

- pastingNg

プログラムによるアクセスに使用されるWijmo **pasting**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pasting** Wijmoイベント名を使用してください。

型 **EventEmitter**

- prepareCellForEditNg

プログラムによるアクセスに使用されるWijmo **prepareCellForEdit**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **prepareCellForEdit** Wijmoイベント名を使用してください。

型 **EventEmitter**

- preserveOutlineState

データがリフレッシュされたときに、グリッドがノードの展開/折りたたみ状態を保持するかどうかを決定する値を取得または設定します。

preserveOutlineState プロパティの実装は、JavaScriptの**Map** オブジェクトに基づいています。このオブジェクトはIE 9およびIE 10で利用できません。

継承元 **FlexGrid**
型 **boolean**

- preserveSelectedState

データがリフレッシュされたときに、グリッドが行の選択状態を保持するかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

- quickAutoSize

グリッドが列のサイズを自動調整するときに精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、自動サイズ変更の機能が無効になります。このプロパティをtrueに設定すると、各列の**quickAutoSize** プロパティの値に従って、その機能が有効になります。null（デフォルト値）に設定した場合、カスタムの**itemFormatter** を持たないグリッドの機能、または**itemFormatter** イベントにアタッチされたハンドラの機能が有効になります。

継承元 **FlexGrid**
型 **boolean**

resizedColumnNg

プログラムによるアクセスに使用されるWijmo **resizedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizedRowNg

プログラムによるアクセスに使用されるWijmo **resizedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizingColumnNg

プログラムによるアクセスに使用されるWijmo **resizingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizingRowNg

プログラムによるアクセスに使用されるWijmo **resizingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

rowAddedNg

プログラムによるアクセスに使用されるWijmo **rowAdded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowAdded** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditEndedNg

プログラムによるアクセスに使用されるWijmo **rowEditEnded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditEnded** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditEndingNg

プログラムによるアクセスに使用されるWijmo **rowEditEnding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditEnding** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditStartedNg

プログラムによるアクセスに使用されるWijmo **rowEditStarted**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditStarted** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditStartingNg

プログラムによるアクセスに使用されるWijmo **rowEditStarting**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditStarting** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowHeaderPath

行ヘッダーセルを作成するために使用されるプロパティの名前を取得または設定します。

行ヘッダーセルは表示されないし、選択することもできません。アクセシビリティツールと組み合わせて使用することを意図しています。

継承元 **FlexGrid**
型 **string**

rowHeaders

行ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

rows

グリッドの行コレクションを取得します。

継承元 **FlexGrid**
型 **RowCollection**

rowsPerItem

各項目の表示に使用される行数を取得します。

この値は、**layoutDefinition**プロパティの値に基づいて 自動的に計算されます。

継承元 **MultiRow**
型 **number**

- scrollPosition

グリッドのスクロールバーの値を表す**Point** を取得または設定します。

継承元 **FlexGrid**
型 **Point**

- scrollPositionChangedNg

プログラムによるアクセスに使用されるWijmo **scrollPositionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **scrollPositionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- scrollSize

グリッド内容のサイズ（ピクセル単位）を取得します。

継承元 **FlexGrid**
型 **Size**

- selectedItems

現在選択されているデータ項目を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

- selectedRows

現在選択されている行を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

- selection

現在の選択を取得または設定します。

継承元 **FlexGrid**
型 **CellRange**

- selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionChangingNg

プログラムによるアクセスに使用されるWijmo **selectionChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionMode

現在の選択モードを取得または設定します。

継承元 **FlexGrid**
型 **SelectionMode**

showAlternatingRows

グリッドで交互表示行のセルに'wj-alt'クラスを追加するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、CSSを変更しなくても、交互表示行スタイルを無効にできます。

継承元 **FlexGrid**
型 **boolean**

showDropDown

グリッドで、**showDropDown** プロパティがtrueに設定されている列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **FlexGrid**
型 **boolean**

showErrors

グリッドで、検証エラーがあるセルとエラーの説明を含むツールチップに'wj-state-invalid' クラスを追加するかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**itemValidator** または**getError** プロパティを使用して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

showGroups

データグループを区切るためにグリッドにグループ行を挿入するかどうかを決定する値を 取得または設定します。

データグループを作成するには、グリッドの **itemsSource** として使用される**ICollectionView** オブジェクトの**groupDescriptions** プロパティを変更します。

継承元 **FlexGrid**
型 **boolean**

showHeaderCollapseButton

グリッドで、行フィールドの列ヘッダーにソートインジケータを グリッドの列ヘッダーパネルに表示するかどうかを判定する値を 取得または設定します。

ボタンが表示されている場合は、ボタンをクリックすると、 **collapsedHeaders**プロパティの値が切り替わります。

The default value for this property is **false**.

継承元 `MultiRow`
型 **boolean**

showMarquee

現在の選択範囲の周囲にマーキー要素を表示するかどうかを示す値を取得または設定します。

継承元 `FlexGrid`
型 **boolean**

showSelectedHeaders

選択されているヘッダセルを示すためにクラス名を追加するかどうかを示す値を取得または設定します。

継承元 `FlexGrid`
型 `HeadersVisibility`

showSort

グリッドで、列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

ソートは、グリッドの`itemsSource` として使用される`ICollectionView` オブジェクトの`sortDescriptions` プロパティによって制御されます。

継承元 `FlexGrid`
型 **boolean**

sortedColumnNg

プログラムによるアクセスに使用されるWijmo **sortedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **sortedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

sortingColumnNg

プログラムによるアクセスに使用されるWijmo **sortingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **sortingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

- sortRowIndex

ソートインジケーターを表示する列ヘッダパネルの行のインデックスを取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、**columnHeaders** パネルの最後の行がソート行になります。

継承元 **FlexGrid**
型 **number**

- stickyHeaders

ユーザーがドキュメントをスクロールしたときに、列ヘッダーを表示したままにするかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

- topLeftCells

左上のセル（列ヘッダーの左）を含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

- treeIndent

異なるレベルの行グループをオフセットするインデントを取得または設定します。

継承元 **FlexGrid**
型 **number**

- updatedLayoutNg

プログラムによるアクセスに使用されるWijmo **updatedLayout**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatedLayout** Wijmoイベント名を使用してください。

型 **EventEmitter**

- updatedViewNg

プログラムによるアクセスに使用されるWijmo **updatedView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatedView** Wijmoイベント名を使用してください。

型 **EventEmitter**

- updatingLayoutNg

プログラムによるアクセスに使用されるWijmo **updatingLayout**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatingLayout** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatingViewNg

プログラムによるアクセスに使用されるWijmo **updatingView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatingView** Wijmoイベント名を使用してください。

型 **EventEmitter**

validateEdits

検証に失敗した編集をユーザーがコミットしようとしたときに、グリッドを編集モードのままにするかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**getError** メソッドを呼び出して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

viewRange

現在表示されているセルの範囲を取得します。

継承元 **FlexGrid**
型 **CellRange**

virtualizationThreshold

仮想化を有効にするために必要な最小行数、最小列数、またはその両方を取得または設定します。

このプロパティはデフォルトでゼロに設定されます。つまり、仮想化は常に有効ということです。これにより、バインディングパフォーマンスとメモリー必要量が向上しますが、スクロール時のパフォーマンス低下は犠牲になります。

グリッドの行数が少ない場合（約50〜100）、このプロパティを150などのやや高い値に設定することで、スクロールのパフォーマンスを向上させることができます。これにより、仮想化が無効になり連結処理が遅くなりますが、認識されるスクロールのパフォーマンスが向上する可能性があります。たとえば、以下のコードは、データソースに150を超える項目がある場合、グリッドがセルを仮想化します。

```
// 150を超える項目がある場合はグリッドを仮想化します
theGrid.virtualizationThreshold = 150;
```

このプロパティを200より大きい値に設定することは推奨されません。ロード処理の時間が長くなり、グリッドはすべての行のセルを作成しているときに数秒間フリーズするため、ページ上の要素数が多いためブラウザが遅くなります。

行と列に別々の仮想化しきい値を設定する場合は、**virtualizationThreshold** プロパティを2つの数値を含む配列に設定できます。この場合、最初の数値は行のしきい値として使用され、2番目の数値は列のしきい値として使用されます。たとえば、次のコードでは、グリッドは行を仮想化しますが、列を仮想化しません。

```
// 行（しきい値0）は仮想化しますが、列は仮想化しません（しきい値10,000）
theGrid.virtualizationThreshold = [0, 10000];
```

継承元 **FlexGrid**
型 **any**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 `Control`
戻り値 **HTMLElement**

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

列をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **c**: **number**
  サイズ変更する列のインデックス。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 `FlexGrid`
戻り値 **void**

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

列の範囲をその内容がちょうど収まるようにサイズ変更します。

測定される行の範囲は常に、現在表示されているすべての行 + 現在表示されていない最大2,000行です。グリッドに大量のデータが含まれている場合には（たとえば、50,000行など）、すべての行を測定すると非常に時間がかかる可能性があるため、すべての行は測定されません。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstColumn**: **number** OPTIONAL
  サイズ変更する最初の列のインデックス（デフォルトは最初の列）。
- **lastColumn**: **number** OPTIONAL
  サイズ変更する最後の列のインデックス（デフォルトは最後の列）。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

行をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **r**: **number**
  サイズ変更する行のインデックス。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の列であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

行の範囲をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックス。
- **lastRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックスす。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の行であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

canEditCell

```
canEditCell(r: number, c: number): void
```

指定されたセルが編集可能かどうかを示す値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

継承元 **FlexGrid**
戻り値 **void**

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

すべてのグループ行を指定したレベルに折りたたみます。

パラメーター

- **level**: **number**
  表示する最大のグループレベル。

継承元 **FlexGrid**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

継承元 **FlexGrid**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

継承元 **FlexGrid**
戻り値 **boolean**

focus

```
focus(): void
```

グリッド全体をスクロールしてビューに入れることなくグリッドにフォーカスを設定するようにオーバーライドされます。

継承元 **FlexGrid**
戻り値 **void**

getBindingColumn

```
getBindingColumn(p: GridPanel, r: number, c: number): Column
```

データ項目をグリッドセルに連結するために使用される**Column** オブジェクトを取得します。

パラメーター

- **p**: **GridPanel**
  セルが含まれる**GridPanel**。
- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

継承元 **MultiRow**
戻り値 **Column**

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セル要素の範囲（ビューポート座標単位）を取得します。

このメソッドは、**cells** パネル内のセル（スクロール可能なデータセル）の範囲を返します。その他のパネルにあるセルの範囲を取得するには、該当する**GridPanel** オブジェクトの**getCellBoundingRect** メソッドを使用してください。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

継承元 **FlexGrid**
戻り値 **Rect**

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

グリッドのスクロール可能領域内のセルに格納された値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

継承元 **FlexGrid**
戻り値 **any**

getClipString

```
getClipString(rng?: CellRange): string
```

CellRange の内容を、クリップボードへのコピーに適した文字列として取得します。

非表示の行および列はクリップボード文字列に含まれません。

パラメーター

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **string**

getColumn

```
getColumn(name: string): Column
```

名前または連結に基づいて列を取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **string**
  検索する名前または連結。

継承元 **MultiRow**
戻り値 **Column**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

GridPanel 内のセルの結合範囲を示す**CellRange** を取得します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。

- **r**: **number**
  セルを含む行のインデックス。

- **c**: **number**
  セルを含む列のインデックス。

- **clip**: **boolean** OPTIONAL
  結合範囲をグリッドの現在のビュー範囲にクリップするかどうか。

継承元 **FlexGrid**
戻り値 **CellRange**

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

パラメーター

- **r**: **number**
  検査するセルの行インデックス。

- **c**: **number**
  検査するセルの列インデックス。

継承元 **FlexGrid**
戻り値 **SelectedState**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

指定されたポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

次に例を示します。

```
// ユーザーがグリッドをクリックしたときに、ポイントをヒットテストします
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

パラメーター

- **pt**: **any**
  調べる**Point** （ページ座標単位）、MouseEventオブジェクト、またはポイントのX座標。

- **y**: **any** OPTIONAL
  ポイントのY座標（ページ座標単位、最初のパラメーターが数値の場合）。

継承元 **FlexGrid**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

指定したCellRangeがこのグリッドの行および列コレクションに対して有効かどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックする範囲。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

autoSizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

autoSizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

autoSizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

autoSizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

beginningEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

cellEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

cellEditEnding イベントを発生させます。

パラメーター

- **e**: **CellEditEndingEventArgs**
  イベントデータを含む**CellEditEndingEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCollapsedHeadersChanged

```
onCollapsedHeadersChanged(e?: EventArgs): void
```

collapsedHeadersChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **MultiRow**
戻り値 **void**

onCollapsedHeadersChanging

```
onCollapsedHeadersChanging(e: CancelEventArgs): boolean
```

collapsedHeadersChanging イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **MultiRow**
戻り値 **boolean**

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

copied イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

copying イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

deletedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

deletingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

draggedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

draggedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

draggingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

draggingColumnOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

draggingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

draggingRowOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

groupCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

groupCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onItemsSourceChanging

```
onItemsSourceChanging(e: CancelEventArgs): boolean
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

loadedRows イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

loadingRows イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

pasted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

pastedCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

pasting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

pastingCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

prepareCellForEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

resizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

resizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

resizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

resizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

rowAdded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

rowEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

rowEditEnding イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

rowEditStarted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

rowEditStarting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

scrollPositionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

selectionChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

sortedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

sortingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

updatedLayout イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

updatingLayout イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

updatingView イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

refresh

```
refresh(fullUpdate?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

継承元 **FlexGrid**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean**
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

- **recycle**: **boolean** OPTIONAL
  既存の要素を再利用するかどうか。

- **state**: **boolean** OPTIONAL
  既存の要素を保持してその状態を更新するかどうか。

継承元 **FlexGrid**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

scrollIntoView

```
scrollIntoView(r: number, c: number, refresh?: boolean): boolean
```

指定したセルが画面に入るようにグリッドをスクロールします。

負の行と列のインデックスは無視されるので、以下を呼び出すと、

```
grid.scrollIntoView(200, -1);
```

グリッドは200行目を表示するように垂直にスクロールしますが、水平にスクロールしません。

パラメーター

- **r**: **number**
  画面に入るようにスクロールする行のインデックス
- **c**: **number**
  画面に入るようにスクロールする列のインデックス。
- **refresh**: **boolean** OPTIONAL
  スクロールした新しい位置をすぐ表示するためにグリッドを更新する必要があるかどうかを決定するオプションのパラメータ。

継承元 **FlexGrid**
戻り値 **boolean**

select

```
select(rng: any, show?: any): void
```

セル範囲を選択し、オプションでそのセル範囲が画面に入るようにスクロールします。

select メソッドは、**CellRange** 、と新しい選択範囲を画面に入るようにスクロールするかどうかを示すオプションのブール型パラメータを渡すことで呼び出すことができます。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(new CellRange(1, 1), true);

// 選択範囲（1,1）~（2,4）を指定し、画面に入るようにスクロールしません。
grid.select(new CellRange(1, 1, 2, 4), false);
```

select メソッドを呼び出してインデックスまたは選択する行と列を渡すこともできます。この場合、選択範囲が画面に入るようにスクロールします。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(1, 1);
```

パラメーター

- **rng**: **any**
  選択する範囲。
- **show**: **any** OPTIONAL
  新しい選択範囲が画面に入るようにスクロールするかどうか。

継承元 **FlexGrid**
戻り値 **void**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

グリッドのスクロール可能領域内のセルの値を設定します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **value**: **any**
  セルに格納する値。
- **coerce**: **boolean** OPTIONAL
  列のデータ型に合わせて値を自動的に変更するかどうか。
- **invalidate**: **boolean** OPTIONAL
  グリッドを無効にして変更を表示するかどうか。

継承元 **FlexGrid**
戻り値 **boolean**

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

文字列を行および列に解析し、その内容を特定の範囲に適用します。

非表示の行および列はスキップされます。

パラメーター

- **text**: **string**
  グリッドに解析する、タブと改行で区切られたテキスト。

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **void**

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

指定されたセルの編集を開始します。

FlexGrid の編集は、Excelの編集によく似ています。 ［F2］キーを押すか、セルをダブルクリックすると、 グリッドが**完全編集** モードになります。 このモードでは、ユーザーが［Enter］、［Tab］、または［Esc］キーを押すか、マウスを使用して選択範囲を移動するまで、セルエディタはアクティブのままになります。完全編集モードでは、カーソルキーを押しても、グリッドの編集モードは終了しません。

テキストをセルに直接入力すると、グリッドは**クイック編集**モードになります。 このモードでは、ユーザーがEnter、Tab、Esc、またはいずれかの矢印キーを押すまで、セルエディタはアクティブなままになります。

通常、完全編集モードは既存の値を変更する際に使用します。 クイック編集モードは新しいデータをすばやく入力する際に使用します。

編集中に［F2］キーを押すことで、完全編集モードとクイック編集モードを切り替えることができます。

パラメーター

- **fullEdit**: **boolean** OPTIONAL
  ユーザーがカーソルキーを押したときも編集モードのままにするかどうか。 デフォルトはtrueです。

- **r**: **number** OPTIONAL
  編集する行のインデックス。 デフォルトは現在選択されている行です。

- **c**: **number** OPTIONAL
  編集する列のインデックス。 デフォルトは現在選択されている列です。

- **focus**: **boolean** OPTIONAL
  編集開始時に、エディタにフォーカスを与えるかどうか。 デフォルトはtrueです。

継承元 **FlexGrid**
戻り値 **boolean**

toggleDropDownList

```
toggleDropDownList(): void
```

現在選択されているセルに関連付けられたドロップダウンリストボックスを切り替えます。

このメソッドでは、セルが編集モードに入ったとき、またはユーザが特定のキーを押したときに、ドロップダウンリストを自動的に表示することができます。

たとえば、このコードでは、グリッドが編集モードに入るたびに、グリッドにドロップダウンリストが表示されます。

```
// グリッドが編集モードに入ると、ドロップダウンリストを表示する。
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

このコードでは、ユーザーがスペースバーを押した場合、グリッドが編集モードに入った後で、ドロップダウンリストが表示されます。

```
// ユーザーがスペースバーを押したときにドロップダウンリストを表示する。
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

継承元 **FlexGrid**
戻り値 **void**

イベント

autoSizedColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizedRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

beginningEdit

セルが編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnded

セル編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnding

セル編集が終了するときに発生します。

このイベントを使用して検証を実行し、無効な編集を防ぐことができます。 たとえば、次のコードは、ユーザーが文字「a」を含まない値を入力できないようにします。このコードは、編集が適用される前に、編集前と編集後の値を取得する方法を示しています。

```
function cellEditEnding (sender, e) {

  // 編集前と編集後の値を取得します
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // newValに「a」が含まれていない場合は、編集をキャンセルします
  e.cancel = newVal.indexOf('a') < 0;
}
```

cancel パラメータをtrueに設定すると、編集された値が無視されて、グリッドでセルの元の値が維持されます。

また、**stayInEditMode** パラメータをtrueに設定すると、グリッドの編集モードが維持され、編集をコミットする前にユーザーが無効な値を修正できます。

継承元 **FlexGrid**
引数 **CellEditEndingEventArgs**

collapsedHeadersChanged

collapsedHeaders プロパティの値が変化した後に発生します。

継承元 **MultiRow**
引数 **EventArgs**

collapsedHeadersChanging

collapsedHeaders プロパティの値が変化した後に発生します。

継承元 **MultiRow**
引数 **CancelEventArgs**

copied

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーした後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

copying

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーするときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラでコピー操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletedRow

ユーザーが［Del］キーを押して行を削除した後に発生します （**allowDelete** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletingRow

ユーザーが［Delete］キーを押して選択されている行を削除するときに発生します（**allowDelete** プロパティを参照）。

イベントハンドラで行の削除をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedColumn

ユーザーが列のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedRow

ユーザーが行のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumn

ユーザーが列のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumnOver

ユーザーが列を別の位置にドラッグするときに発生します。

ハンドラは、このイベントをキャンセルして、ユーザーが特定の位置に列をドロップしないようにすることができます。次に例を示します。

```
// ドラッグされている列を記憶します
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// 'sales'列がインデックス0にドラッグされないようにします
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRow

ユーザーが行のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRowOver

ユーザーが行を別の位置にドラッグするときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

formatItem

セルを表す要素が作成されたときに発生します。

このイベントを使用してセルを表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、グループ行のセルから'wj-wrap'クラスを削除します。

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

継承元 **FlexGrid**
引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

groupCollapsedChanged

グループが展開または折りたたまれた後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

groupCollapsedChanging

グループが展開または折りたたまれる直前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

itemsSourceChanged

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

itemsSourceChanging

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

loadedRows

グリッド行がデータソースの項目に連結された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

loadingRows

グリッド行がデータソースの項目に連結される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

pasted

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastedCell

ユーザーがクリップボードの内容をセルに貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pasting

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた時に発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastingCell

ユーザーがクリップボードの内容をセルに貼り付けるときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

prepareCellForEdit

エディタセルが作成されたとき、それがアクティブになる前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

resizedColumn

ユーザーが列のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizedRow

ユーザーが行のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingColumn

列がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingRow

行がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowAdded

ユーザーが新規行テンプレートを編集して新しい項目を作成したときに発生します（**allowAddNew** プロパティを参照）。

イベントハンドラで新しい項目の内容をカスタマイズしたり、新しい項目の作成をキャンセルしたりできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnded

行編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnding

行編集が終了するとき、変更が確定またはキャンセルされる前に発生します。

このイベントを**rowEditStarted**イベントと組み合わせて使用して、ディープ連結編集を元に戻す操作を実装できます。次に例を示します。

```
// 編集の開始時にディープ連結値を保存します
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.')> -1) { // ディープ連結
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// 編集がキャンセルされたときにディープ連結値を復元します
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // ユーザーによって編集がキャンセルされました
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarted

行が編集モードに入った後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarting

行が編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

scrollPositionChanged

コントロールがスクロールされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

selectionChanged

選択が変更された後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

selectionChanging

選択が変更される前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortedColumn

ユーザーが列ヘッダをクリックしてソートを適用した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortingColumn

ユーザーが列ヘッダをクリックしてソートを適用する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

updatedLayout

グリッドで内部レイアウトが更新された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

updatedView

グリッドが現在のビューを構成する要素の作成/更新を完了したときに発生します。

グリッドのビューは以下のような操作に応じて更新されます。

- グリッドまたはそのデータソースの更新
- 行または列の追加、削除、変更
- グリッドのサイズ変更またはスクロール
- 選択の変更

継承元 **FlexGrid**
引数 **EventArgs**

updatingLayout

グリッドで内部レイアウトが更新される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

updatingView

現在のビューを構成する要素の作成/更新をグリッドが開始したときに発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

wijmo/wijmo.angular2.grid.sheet モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.grid.sheet**

wijmo.grid.sheetモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.grid.sheetは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjSheet from 'wijmo/wijmo.angular2.grid.sheet';

@Component({
    directives: [wjSheet.WjFlexSheet],
    template: `<wj-flex-sheet></wj-flex-sheet>`,
    selector: 'my-cmp',
})
export class MyCmp {
}
```

クラス

- WjFlexSheet
- WjSheet

WjFlexSheet クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid.sheet**
基本クラス **FlexSheet**
表示 ☐継承されたメンバー ☐イベント発生元

FlexSheet コントロールに対応するAngular 2コンポーネント。

wj-flex-sheetコンポーネントを使用して、Angular 2アプリケーションに**FlexSheet**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexSheetコンポーネントは、**FlexSheet**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-sheet**コンポーネントには、**WjSheet** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- asyncBindings
- autoClipboard
- autoGenerateColumns
- autoScroll
- autoSearch
- autoSizedColumnNg
- autoSizedRowNg
- autoSizeMode
- autoSizingColumnNg
- autoSizingRowNg
- beginningEditNg
- bottomLeftCells
- cellEditEndedNg
- cellEditEndingNg
- cellFactory
- cells
- childItemsPath
- clientSize
- cloneFrozenCells
- collectionView
- columnChangedNg
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect

- controlTemplate
- copiedNg
- copyingNg
- deferResizing
- definedNames
- deletedRowNg
- deletingRowNg
- draggedColumnNg
- draggedRowNg
- draggingColumnNg
- draggingColumnOverNg
- draggingRowColumnNg
- draggingRowNg
- draggingRowOverNg
- droppingRowColumnNg
- editableCollectionView
- editRange
- enableDragDrop
- enableFormulas
- filter
- formatItemNg
- frozenColumns
- frozenRows
- gotFocusNg
- groupCollapsedChangedNg
- groupCollapsedChangingNg
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- initialized
- isDisabled
- isFunctionListOpen
- isInitialized
- isReadOnly
- isTabHolderVisible
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceChangedNg
- itemValidator
- keyActionEnter
- keyActionTab
- loadedNg
- loadedRowsNg
- loadingRowsNg
- lostFocusNg
- mergeManager

- newRowAtTop
- pastedCellNg
- pastedNg
- pastingCellNg
- pastingNg
- prepareCellForEditNg
- prepareChangingColumnNg
- prepareChangingRowNg
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- resizedColumnNg
- resizedRowNg
- resizingColumnNg
- resizingRowNg
- rightToLeft
- rowAddedNg
- rowChangedNg
- rowEditEndedNg
- rowEditEndingNg
- rowEditStartedNg
- rowEditStartingNg
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollPositionChangedNg
- scrollSize
- selectedItems
- selectedRows
- selectedSheet
- selectedSheetChangedNg
- selectedSheetIndex
- selection
- selectionChangedNg
- selectionChangingNg
- selectionMode
- sheetClearedNg
- sheets
- showAlternatingRows
- showDropDown
- showErrors
- showFilterIcons
- showGroups
- showMarquee
- showSelectedHeaders
- showSort
- sortedColumnNg
- sortingColumnNg

- sortingColumnNg
- sortManager
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- undoStack
- unknownFunctionNg
- updatedLayoutNg
- updatedViewNg
- updatingLayoutNg
- updatingViewNg
- validateEdits
- viewRange
- virtualizationThreshold
- wjModelProperty

メソッド

- addBoundSheet
- addEventListener
- addFunction
- addUnboundSheet
- applyCellsStyle
- applyFunctionToCell
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- clear
- collapseGroupsToLevel
- containsFocus
- convertNumberToAlpha
- created
- deferUpdate
- deleteColumns
- deleteRows
- dispose
- disposeAll
- endUpdate
- evaluate
- finishEditing
- focus
- freezeAtCursor
- getBuiltInTableStyle
- getCellBoundingRect
- getCellData

- getCellValue
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getSelectionFormatState
- getTemplate
- hideFunctionList
- hitTest
- initialize
- insertColumns
- insertRows
- invalidate
- invalidateAll
- isRangeValid
- load
- loadAsync
- mergeRange
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onColumnChanged
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowColumn
- onDraggingRowOver
- onDroppingRowColumn
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onItemsSourceChanging
- onLoaded
- onLoadedRows
- onLoadingRows
- onLostFocus

- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onPrepareChangingColumn
- onPrepareChangingRow
- onRefreshed
- onRefreshing
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowChanged
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectedSheetChanged
- onSelectionChanged
- onSelectionChanging
- onSheetCleared
- onSortedColumn
- onSortingColumn
- onUnknownFunction
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- redo
- refresh
- refreshAll
- refreshCells
- removeEventListener
- save
- saveAsync
- scrollIntoView
- select
- selectNextFunction
- selectPreviousFunction
- setCellData
- setClipString
- showColumnFilter
- showFunctionList
- startEditing
- toggleDropDownList
- undo

イベント

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- columnChanged
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowColumn
- draggingRowOver
- droppingRowColumn
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- itemsSourceChanging
- loaded
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- prepareChangingColumn
- prepareChangingRow
- refreshed
- refreshing
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowChanged
- rowEditEnded
- rowEditEnding

- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectedSheetChanged
- selectionChanged
- selectionChanging
- sheetCleared
- sortedColumn
- sortingColumn
- unknownFunction
- updatedLayout
- updatedView
- updatingLayout
- updatingView

コンストラクタ

constructor

```
constructor(element: any, options?): FlexSheet
```

Initializes a new instance of the **FlexSheet** class.

パラメーター

- **element**: **any**
  The DOM element that hosts the control, or a CSS selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

継承元 **FlexSheet**
戻り値 **FlexSheet**

プロパティ

activeEditor

現在アクティブになっているセルエディタを表す**HTMLInputElement**を取得します。

継承元 **FlexGrid**
型 **HTMLInputElement**

allowAddNew

ユーザーがソースコレクションに項目を追加できるようにグリッドに新規行テンプレートを表示するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、新規行テンプレートは表示されません。

継承元 **FlexGrid**
型 **boolean**

allowDelete

ユーザーが［Delete］キーを押したときにグリッドの選択されている行を削除するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、選択されている行は削除されません。

継承元 **FlexGrid**
型 **boolean**

allowDragging

ユーザーがマウスを使用して行、列、またはその両方をドラッグできるかどうかを決定する値を取得または設定します。

autoScroll プロパティがtrueに設定されている場合、ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされます。

グリッドでは、列のドラッグがデフォルトで有効になっています。

グリッドが連結モードでは、行をドラッグするには特別な 考慮が必要になります。

グリッドが連結モードでは、行をドラッグするとデータソースとの同期が失われます（たとえば行4は6行として参照されることがあります）。 これを回避するには、**draggedRow** イベントを処理し、データを新しい行の位置と同期させる必要があります。

また、**allowSorting** プロパティをfalseに設定する必要があります。そうしないと、行の順序がデータによって決定され、行をドラッグするのは無意味になります。

次のフィドルでは、連結グリッドでの行のドラッグを実行しています。 **Bound Row Dragging (http://jsfiddle.net/Wijmo5/kyg0qsda/)**.

継承元 **FlexGrid**
型 **AllowDragging**

allowMerging

グリッドのどの部分のセル結合を許可するかを取得または設定します。

継承元 **FlexGrid**
型 **AllowMerging**

allowResizing

ユーザーがマウスを使用して行、列、またはその両方をサイズ変更できるかどうかを決定する値を取得または設定します。

サイズ変更が可能な場合は、列ヘッダーセルの右端をドラッグして列を、行ヘッダーセルの下端をドラッグして行をサイズ変更できます。

ヘッダーセルの端をダブルクリックして、行および列をコンテンツに合わせて自動的にサイズ変更することもできます。 自動サイズ変更の動作は、**autoSizeMode** プロパティを使用してカスタマイズできます。

継承元 **FlexGrid**
型 **AllowResizing**

allowSorting

ユーザーが列ヘッダーセルをクリックして列をソートできるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoClipboard

グリッドがクリップボードショートカットを処理するかどうかを決定する値を取得または設定します。

次のクリップボードショートカットがあります。

［Ctrl］+［C］、［Ctrl］+［Ins］	グリッドの選択範囲をクリップボードにコピーします。
［Ctrl］+［V］、［Shift］+［Ins］	クリップボードのテキストをグリッドの選択範囲に貼り付けます。

クリップボード操作は、表示されている行および列だけが対象となります。

読み取り専用のセルは、貼り付け操作の影響を受けません。

継承元 **FlexGrid**
型 **boolean**

autoGenerateColumns

グリッドが**itemsSource** に基づいて列を自動的に生成するかどうかを 決定する値を取得または設定します。

列の生成は、少なくとも1項目を含む**itemsSource** プロパティに依存します。このデータ項目が検査され、列が作成されて、プリミティブ値（数値、文字列、Boolean、またはDate）を含むプロパティに連結されます。

プロパティをnullに設定すると、適切なタイプを推測する方法がないため、列は生成されません。このような場合は、**autoGenerateColumns** プロパティをfalseに設定し、明示的に列を作成する必要があります。次に例を示します。

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // データ項目にnull値が含まれる場合があります
  columns: [                  // そのため、列を明示的に定義します
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

継承元 **FlexGrid**
型 **boolean**

autoScroll

ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされるかどうかを決定する値を取得または設定します。

行と列のドラッグは、**allowDragging** プロパティによって制御されます。

このプロパティはデフォルトでtrueに設定されます。

継承元 **FlexGrid**
型 **boolean**

autoSearch

ユーザーが読み取り専用セルに入力するに伴ってグリッドでセルを検索するかどうかを決定する値を取得または設定します。

検索が現在選択されている列(編集不可能な場合)で行われます。検索は現在選択されている行から始まり、大文字と小文字を区別されません。

継承元 `FlexGrid`
型 `boolean`

autoSizedColumnNg

プログラムによるアクセスに使用されるWijmo **autoSizedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizedColumn** Wijmoイベント名を使用してください。

型 `EventEmitter`

autoSizedRowNg

プログラムによるアクセスに使用されるWijmo **autoSizedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizedRow** Wijmoイベント名を使用してください。

型 `EventEmitter`

autoSizeMode

行または列のサイズを自動設定するときにどのセルを考慮に入れるかを取得または設定します。

このプロパティは、ユーザーが列ヘッダの端をダブルクリックしたときの動作を制御します。

デフォルトでは、その列のヘッダセルとデータセルの内容に基づいて列の幅が自動的に設定されます。このプロパティを使用すると、ヘッダのみまたはデータのみに基づいて列の幅を設定するように変更できます。

継承元 `FlexGrid`
型 `AutoSizeMode`

autoSizingColumnNg

プログラムによるアクセスに使用されるWijmo **autoSizingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizingColumn** Wijmoイベント名を使用してください。

型 `EventEmitter`

autoSizingRowNg

プログラムによるアクセスに使用されるWijmo **autoSizingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizingRow** Wijmoイベント名を使用してください。

型 `EventEmitter`

beginningEditNg

プログラムによるアクセスに使用されるWijmo **beginningEdit**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **beginningEdit** Wijmoイベント名を使用してください。

型 **EventEmitter**

bottomLeftCells

左下のセルを含む**GridPanel** を取得します。

bottomLeftCells パネルは、行ヘッダの下、**columnFooters** パネルの左に表示されます。

継承元 **FlexGrid**
型 **GridPanel**

cellEditEndedNg

プログラムによるアクセスに使用されるWijmo **cellEditEnded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **cellEditEnded** Wijmoイベント名を使用してください。

型 **EventEmitter**

cellEditEndingNg

プログラムによるアクセスに使用されるWijmo **cellEditEnding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **cellEditEnding** Wijmoイベント名を使用してください。

型 **EventEmitter**

cellFactory

このグリッドのセルを作成および更新する**CellFactory** を取得または設定します。

継承元 **FlexGrid**
型 **CellFactory**

cells

データセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

childItemsPath

階層グリッドで子の行の生成に使用される 1 つ以上のプロパティの名前を取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `childItemsPath = 'items'` ）。

項目の異なるレベルに異なる名前を持つ子項目がある場合は、このプロパティに、各レベルの子項目を含むプロパティの名前から成る配列を設定します。（たとえば、 `childItemsPath = ['checks','earnings'];` ）。

サンプル

Show me (https://jsfiddle.net/Wijmo5/kk9j93bL)

継承元 **FlexGrid**
型 **any**

clientSize

コントロールのクライアントサイズを取得します（コントロールのサイズからヘッダーとスクロールバーを引いた値）。

継承元 **FlexGrid**
型 **Size**

cloneFrozenCells

スクロール中に知覚されるパフォーマンスを向上させるには、FlexGridがフリーズされたセルを複製し、別の要素に表示するかどうかを決定する値を取得または設定します。

このプロパティのデフォルト値はnullであり、ブラウザーによって一番適当な設定が選択されます。

継承元 **FlexGrid**
型 **boolean**

collectionView

グリッドデータを含む**ICollectionView** を取得します。

継承元 **FlexGrid**
型 **ICollectionView**

columnChangedNg

プログラムによるアクセスに使用されるWijmo **columnChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **columnChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

columnFooters

列フッターセルを保持する**GridPanel** を取得します。

columnFooters パネルは、グリッドセルの下、**bottomLeftCells** パネルの右に表示されます。これを使用して、グリッドデータの下にサマリー情報を表示できます。

以下の例では、**columnFooters** パネルに 1行を追加して、**aggregate** プロパティが設定された列に サマリーデータを表示する方法を示します。

```
function addFooterRow(flex) {

  // 集計を表示するGroupRowを作成します
  var row = new wijmo.grid.GroupRow();

  // 列フッターパネルに行を追加します
  flex.columnFooters.rows.push(row);

  // ヘッダーにシグマを表示します
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

継承元 **FlexGrid**
型 **GridPanel**

columnHeaders

列ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

columnLayout

現在の列レイアウトを定義するJSON文字列を取得または設定します。

列レイアウト文字列は、列とそのプロパティを含む配列を表します。これを使用して、ユーザーが定義した列レイアウトをセッションを越えて保持できます。また、ユーザーが列レイアウトを変更できるアプリケーションで元に戻す/やり直し機能を実装することもできます。

データマップはシリアル化できないため、列レイアウト文字列に**dataMap** プロパティは含まれていません。

継承元 **FlexGrid**
型 **string**

columns

グリッドの列コレクションを取得します。

継承元 **FlexGrid**
型 **ColumnCollection**

controlRect

コントロールの外接矩形（ページ座標単位）を取得します。

継承元 **FlexGrid**
型 **Rect**

STATIC controlTemplate

Overrides the template used to instantiate **`FlexSheet`** control.

継承元 **`FlexSheet`**
型 **any**

copiedNg

プログラムによるアクセスに使用されるWijmo **copied**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **copied** Wijmoイベント名を使用してください。

型 **EventEmitter**

copyingNg

プログラムによるアクセスに使用されるWijmo **copying**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **copying** Wijmoイベント名を使用してください。

型 **EventEmitter**

deferResizing

ユーザーがマウスボタンを放すまで、行および列のサイズ変更を遅らせるかどうかを決定する値を取得または設定します。

デフォルトでは、**`deferResizing`** はfalseに設定されており、ユーザーがマウスをドラッグするに伴って行および列がサイズ変更されます。このプロパティをtrueに設定すると、グリッドにサイズ変更マーカーが表示され、ユーザーがマウスボタンを放したときに初めて行または列のサイズが変更されます。

継承元 **`FlexGrid`**
型 **boolean**

definedNames

Gets an array the **`IDefinedName`** objects representing named ranges/expressions defined in the **FlexSheet**.

継承元 **`FlexSheet`**
型 **`ObservableArray`**

deletedRowNg

プログラムによるアクセスに使用されるWijmo **deletedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **deletedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

deletingRowNg

プログラムによるアクセスに使用されるWijmo **deletingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **deletingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggedColumnNg

プログラムによるアクセスに使用されるWijmo **draggedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggedRowNg

プログラムによるアクセスに使用されるWijmo **draggedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingColumnNg

プログラムによるアクセスに使用されるWijmo **draggingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingColumnOverNg

プログラムによるアクセスに使用されるWijmo **draggingColumnOver**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingColumnOver** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingRowColumnNg

プログラムによるアクセスに使用されるWijmo **draggingRowColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingRowColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingRowNg

プログラムによるアクセスに使用されるWijmo **draggingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingRowOverNg

プログラムによるアクセスに使用されるWijmo **draggingRowOver**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingRowOver** Wijmoイベント名を使用してください。

型 **EventEmitter**

- droppingRowColumnNg

プログラムによるアクセスに使用されるWijmo **droppingRowColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **droppingRowColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

- editableCollectionView

グリッドデータを含む**IEditableCollectionView** を取得します。

継承元 **FlexGrid**
型 **IEditableCollectionView**

- editRange

現在編集中のセルを識別する**CellRange** を取得します。

継承元 **FlexGrid**
型 **CellRange**

- enableDragDrop

Gets or sets the value to indicates whether enable drag and drop rows or columns in FlexSheet.

継承元 **FlexSheet**
型 **boolean**

- enableFormulas

Gets or sets the value to indicates whether enable formulas in FlexSheet.

継承元 **FlexSheet**
型 **boolean**

- filter

Gets the **FlexSheetFilter** instance that controls **FlexSheet** filtering.

継承元 **FlexSheet**
型 **FlexSheetFilter**

- formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

- frozenColumns

固定された列の数を取得または設定します。

固定された列は水平方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

- frozenRows

静止行の数を取得または設定します。

固定された行は垂直方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- groupCollapsedChangedNg

プログラムによるアクセスに使用されるWijmo **groupCollapsedChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **groupCollapsedChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- groupCollapsedChangingNg

プログラムによるアクセスに使用されるWijmo **groupCollapsedChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **groupCollapsedChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupHeaderFormat

グループヘッダーコンテンツを作成するために使用される書式文字列を取得または設定します。

この文字列は、任意のテキストと次の置換文字列を含むことができます。

- **{name}**：グループ化されるプロパティの名前。
- **{value}**：グループ化されるプロパティの値。
- **{level}**：グループレベル。
- **{count}**：このグループ内にある項目の合計数。

列がグループ化プロパティに連結されている場合は、列ヘッダーを使用して パラメータを置換し、列の書式とデータマップを使用して `{value}` パラメータを計算します。 列がない場合は、代わりにグループ情報が使用されます。

グループプロパティに連結された非表示の列を追加して、グループヘッダーセルの書式設定をカスタマイズすることもできます。

このプロパティのデフォルト値は
`'{name}: <b>{value}</b>({count:n0} items)'` です。 これは、 `'Country: UK (12 items)'` や
`'Country: Japan (8 items)'` のようなグループヘッダーを作成します。

継承元 **FlexGrid**
型 **string**

headersVisibility

行ヘッダおよび列ヘッダが表示されるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

imeEnabled

編集モードでないときに、グリッドがIME（Input Method Editor）をサポートするかどうかを決定する値を取得または設定します。

このプロパティは、日本語、中国語、韓国語など、IMEサポートを必要とする言語のサイト/アプリケーションにのみ関係します。

継承元 **FlexGrid**
型 **boolean**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isFunctionListOpen

Gets a value indicating whether the function list is opened.

継承元 **FlexSheet**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用してセル値を変更できるかどうかを判定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

- isTabHolderVisible

Gets or sets a value indicating whether the TabHolder is visible.

継承元 **FlexSheet**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

このグリッドのセルのカスタマイズに使用されるフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意のセルに任意の内容を追加できます。そのため、グリッドセルの外観と動作を非常に柔軟に制御することが可能です。

この関数は、セルを含む**GridPanel**、セルの行インデックスと列インデックス、セルを表すHTML要素の4つのパラメーターをとります。通常は、関数内でセル要素の**innerHTML**　プロパティを変更するようにします。

例:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == wijmo.grid.CellType.Cell) {

    // セルにスパークラインを描画します。
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

FlexGridはセルをリサイクルするため、**itemFormatter** によってセルのスタイル属性を変更する場合は、 セルの適切でないスタイル属性を事前にリセットする必要があります。例:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // カスタマイズする属性をリセットします。
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // このセルのcolorおよびbackgroundColor属性をカスタマイズします。
  ...
}
```

複数のクライアントがグリッドのレンダリングをカスタマイズできるようにする場合は（たとえば、ディレクティブや再利用可能なライブラリを作成するときなど）、代わりに**formatItem** イベントを使用することを検討してください。このイベントを使用すると、複数のクライアントがそれぞれ独自のハンドラをアタッチできます。

継承元　　**FlexGrid**
型　　**Function**

itemsSource

グリッドに表示される項目を含む配列または**ICollectionView** を取得または設定します。

継承元　　**FlexGrid**
型　　**any**

itemsSourceChangedNg

プログラムによるアクセスに使用されるWijmo **itemsSourceChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemsSourceChanged** Wijmoイベント名を使用してください。

型　　**EventEmitter**

itemValidator

セルに有効なデータが含まれているかどうかを決定するバリデータ関数を取得または設定します。

指定された場合、バリデータ関数はセルの行インデックスと列インデックスを含む2つのパラメータをとり、エラーの説明を含む文字列を返す必要があります。

このプロパティは、バインドされていないグリッドを処理する場合に特に便利です。バインドされたグリッドは、代わりに**getError** プロパティを使用して検証できます。

この例では、セルのすぐ上のセルと同じデータがセルに含まれないようにする方法を示します。

```
// 上のセルは、現在のセルと同じ値が含まれていないことを確認します
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'これは重複した値です...'
    }
  }
  return null; // エラーなし
}
```

継承元 **FlexGrid**
型 **Function**

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は**MoveDown** となり、コントロールがカーソルを次の行に移動します。 この動作は標準的なExcel式の動作です。

継承元 **FlexGrid**
型 **KeyAction**

keyActionTab

［Tab］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は、**None** となります。これにより、Tabキーが押されたときにブラウザ上で次または前のコントロールを選択できます。これは、ページのアクセシビリティを向上させるために推奨される設定になります。

以前のバージョンでは、デフォルトは**Cycle** に設定されていました。これにより、コントロールがカーソルを、グリッドを横切って下部に移動しました。これは標準的なExcelの動作ですが、アクセシビリティには適していません。

また、**CycleOut** の設定が存在します。これにより、カーソルがセルを通じて移動（**Cycle** ）してから、最後または最初のセルが選択されたときにページの次/前のコントロールに移動します 。

継承元 **FlexGrid**
型 **KeyAction**

loadedNg

プログラムによるアクセスに使用されるWijmo **loaded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loaded** Wijmoイベント名を使用してください。

型 **EventEmitter**

loadedRowsNg

プログラムによるアクセスに使用されるWijmo **loadedRows**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadedRows** Wijmoイベント名を使用してください。

型 **EventEmitter**

loadingRowsNg

プログラムによるアクセスに使用されるWijmo **loadingRows**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadingRows** Wijmoイベント名を使用してください。

型 **EventEmitter**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

mergeManager

セルの結合方法を決定する**MergeManager** オブジェクトを取得または設定します。

継承元 **FlexGrid**
型 **MergeManager**

newRowAtTop

新しい行テンプレートをグリッドの上部に配置するか、下部に配置するかを示す値を取得または設定します。

newRowAtTop プロパティをtrueに設定した場合、新しい行テンプレートを常に表示したままにする場合は、**frozenRows** プロパティを 1に設定します。これにより、新しい行テンプレートは上部に固定され、ビューからスクロールオフされなくなります。

allowAddNew プロパティがtrueに設定されている場合、および**itemsSource** オブジェクトが新しい項目の追加をサポートしている場合にのみ、新しい行テンプレートが表示されます。

継承元 **FlexGrid**
型 **boolean**

pastedCellNg

プログラムによるアクセスに使用されるWijmo **pastedCell**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pastedCell** Wijmoイベント名を使用してください。

型 **EventEmitter**

pastedNg

プログラムによるアクセスに使用されるWijmo **pasted**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pasted** Wijmoイベント名を使用してください。

型 **EventEmitter**

pastingCellNg

プログラムによるアクセスに使用されるWijmo **pastingCell**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pastingCell** Wijmoイベント名を使用してください。

型 **EventEmitter**

pastingNg

プログラムによるアクセスに使用されるWijmo **pasting**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pasting** Wijmoイベント名を使用してください。

型 **EventEmitter**

prepareCellForEditNg

プログラムによるアクセスに使用されるWijmo **prepareCellForEdit**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **prepareCellForEdit** Wijmoイベント名を使用してください。

型 **EventEmitter**

prepareChangingColumnNg

プログラムによるアクセスに使用されるWijmo **prepareChangingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **prepareChangingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

prepareChangingRowNg

プログラムによるアクセスに使用されるWijmo **prepareChangingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **prepareChangingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

preserveOutlineState

データがリフレッシュされたときに、グリッドがノードの展開/折りたたみ状態を保持するかどうかを決定する値を取得または設定します。

`preserveOutlineState` プロパティの実装は、JavaScriptの**Map** オブジェクトに基づいています。このオブジェクトはIE 9およびIE 10で利用できません。

継承元 **`FlexGrid`**
型 **boolean**

preserveSelectedState

データがリフレッシュされたときに、グリッドが行の選択状態を保持するかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

quickAutoSize

グリッドが列のサイズを自動調整するときに精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、自動サイズ変更の機能が無効になります。このプロパティをtrueに設定すると、各列の**quickAutoSize** プロパティの値に従って、その機能が有効になります。null（デフォルト値）に設定した場合、カスタムの**itemFormatter** を持たないグリッドの機能、または**itemFormatter** イベントにアタッチされたハンドラの機能が有効になります。

継承元 **FlexGrid**
型 **boolean**

resizedColumnNg

プログラムによるアクセスに使用されるWijmo **resizedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizedRowNg

プログラムによるアクセスに使用されるWijmo **resizedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizingColumnNg

プログラムによるアクセスに使用されるWijmo **resizingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizingRowNg

プログラムによるアクセスに使用されるWijmo **resizingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rowAddedNg

プログラムによるアクセスに使用されるWijmo **rowAdded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowAdded** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowChangedNg

プログラムによるアクセスに使用されるWijmo **rowChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditEndedNg

プログラムによるアクセスに使用されるWijmo **rowEditEnded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditEnded** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditEndingNg

プログラムによるアクセスに使用されるWijmo **rowEditEnding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditEnding** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditStartedNg

プログラムによるアクセスに使用されるWijmo **rowEditStarted**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditStarted** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditStartingNg

プログラムによるアクセスに使用されるWijmo **rowEditStarting**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditStarting** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowHeaderPath

行ヘッダーセルを作成するために使用されるプロパティの名前を取得または設定します。

行ヘッダーセルは表示されないし、選択することもできません。アクセシビリティツールと組み合わせて使用することを意図しています。

継承元 **FlexGrid**
型 **string**

- rowHeaders

行ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

- rows

グリッドの行コレクションを取得します。

継承元 **FlexGrid**
型 **RowCollection**

- scrollPosition

グリッドのスクロールバーの値を表す**Point** を取得または設定します。

継承元 **FlexGrid**
型 **Point**

- scrollPositionChangedNg

プログラムによるアクセスに使用されるWijmo **scrollPositionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **scrollPositionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- scrollSize

グリッド内容のサイズ（ピクセル単位）を取得します。

継承元 **FlexGrid**
型 **Size**

- selectedItems

現在選択されているデータ項目を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

- selectedRows

現在選択されている行を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは**selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selectedSheet

Gets the current **Sheet** in the **FlexSheet**.

継承元 **FlexSheet**
型 **Sheet**

selectedSheetChangedNg

プログラムによるアクセスに使用されるWijmo **selectedSheetChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedSheetChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectedSheetIndex

Gets or sets the index of the current sheet in the **FlexSheet**.

継承元 **FlexSheet**
型 **number**

selection

現在の選択を取得または設定します。

継承元 **FlexGrid**
型 **CellRange**

selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionChangingNg

プログラムによるアクセスに使用されるWijmo **selectionChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionMode

現在の選択モードを取得または設定します。

継承元 **FlexGrid**
型 **SelectionMode**

sheetClearedNg

プログラムによるアクセスに使用されるWijmo **sheetCleared**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **sheetCleared** Wijmoイベント名を使用してください。

型 **EventEmitter**

sheets

Gets the collection of **Sheet** objects representing workbook sheets.

継承元 **FlexSheet**
型 **SheetCollection**

showAlternatingRows

グリッドで交互表示行のセルに'wj-alt'クラスを追加するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、CSSを変更しなくても、交互表示行スタイルを無効にできます。

継承元 **FlexGrid**
型 **boolean**

showDropDown

グリッドで、**showDropDown** プロパティがtrueに設定されている列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **FlexGrid**
型 **boolean**

showErrors

グリッドで、検証エラーがあるセルとエラーの説明を含むツールチップに'wj-state-invalid' クラスを追加するかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**itemValidator** または**getError** プロパティを使用して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

showFilterIcons

Gets or sets the visiblity of the filter icon.

継承元 **FlexSheet**
型 **boolean**

showGroups

データグループを区切るためにグリッドにグループ行を挿入するかどうかを決定する値を 取得または設定します。

データグループを作成するには、グリッドの **itemsSource** として使用される **ICollectionView** オブジェクトの **groupDescriptions** プロパティを変更します。

継承元 **FlexGrid**
型 **boolean**

showMarquee

現在の選択範囲の周囲にマーキー要素を表示するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

showSelectedHeaders

選択されているヘッダセルを示すためにクラス名を追加するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

showSort

グリッドで、列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

ソートは、グリッドの **itemsSource** として使用される **ICollectionView** オブジェクトの **sortDescriptions** プロパティによって制御されます。

継承元 **FlexGrid**
型 **boolean**

sortedColumnNg

プログラムによるアクセスに使用されるWijmo **sortedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **sortedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

sortingColumnNg

プログラムによるアクセスに使用されるWijmo **sortingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **sortingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

sortManager

Gets the **SortManager** instance that controls **FlexSheet** sorting.

継承元 **FlexSheet**
型 **SortManager**

- sortRowIndex

ソートインジケーターを表示する列ヘッダパネルの行のインデックスを取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、**columnHeaders** パネルの最後の行がソート行になります。

継承元 **FlexGrid**
型 **number**

- stickyHeaders

ユーザーがドキュメントをスクロールしたときに、列ヘッダーを表示したままにするかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

- topLeftCells

左上のセル（列ヘッダーの左）を含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

- treeIndent

異なるレベルの行グループをオフセットするインデントを取得または設定します。

継承元 **FlexGrid**
型 **number**

- undoStack

Gets the **UndoStack** instance that controls undo and redo operations of the **FlexSheet**.

継承元 **FlexSheet**
型 **UndoStack**

- unknownFunctionNg

プログラムによるアクセスに使用されるWijmo **unknownFunction**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **unknownFunction** Wijmoイベント名を使用してください。

型 **EventEmitter**

- updatedLayoutNg

プログラムによるアクセスに使用されるWijmo **updatedLayout**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatedLayout** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatedViewNg

プログラムによるアクセスに使用されるWijmo **updatedView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatedView** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatingLayoutNg

プログラムによるアクセスに使用されるWijmo **updatingLayout**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatingLayout** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatingViewNg

プログラムによるアクセスに使用されるWijmo **updatingView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatingView** Wijmoイベント名を使用してください。

型 **EventEmitter**

validateEdits

検証に失敗した編集をユーザーがコミットしようとしたときに、グリッドを編集モードのままにするかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**`itemsSource`** の**`getError`** メソッドを呼び出して、検証エラーを検出します。

継承元 **`FlexGrid`**
型 **boolean**

viewRange

現在表示されているセルの範囲を取得します。

継承元 **`FlexGrid`**
型 **`CellRange`**

virtualizationThreshold

仮想化を有効にするために必要な最小行数、最小列数、またはその両方を取得または設定します。

このプロパティはデフォルトでゼロに設定されます。つまり、仮想化は常に有効ということです。これにより、バインディングパフォーマンスとメモリー必要量が向上しますが、スクロール時のパフォーマンス低下は犠牲になります。

グリッドの行数が少ない場合（約50〜100）、このプロパティを150などのやや高い値に設定することで、スクロールのパフォーマンスを向上させることができます。これにより、仮想化が無効になり連結処理が遅くなりますが、認識されるスクロールのパフォーマンスが向上する可能性があります。たとえば、以下のコードは、データソースに150を超える項目がある場合、グリッドがセルを仮想化します。

```
// 150を超える項目がある場合はグリッドを仮想化します
theGrid.virtualizationThreshold = 150;
```

このプロパティを200より大きい値に設定することは推奨されません。ロード処理の時間が長くなり、グリッドはすべての行のセルを作成しているときに数秒間フリーズするため、ページ上の要素数が多いためブラウザが遅くなります。

行と列に別々の仮想化しきい値を設定する場合は、**virtualizationThreshold** プロパティを2つの数値を含む配列に設定できます。この場合、最初の数値は行のしきい値として使用され、2番目の数値は列のしきい値として使用されます。たとえば、次のコードでは、グリッドは行を仮想化しますが、列を仮想化しません。

```
// 行（しきい値0）は仮想化しますが、列は仮想化しません（しきい値10,000）
theGrid.virtualizationThreshold = [0, 10000];
```

継承元 **FlexGrid**
型 **any**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は''です。

型 **string**

メソッド

addBoundSheet

```
addBoundSheet(sheetName: string, source: any, pos?: number, grid?: FlexGrid): Sheet
```

Add a bound **Sheet** to the **FlexSheet**.

パラメーター

- **sheetName**: **string**
  The name of the **Sheet**.
- **source**: **any**
  The items source for the **Sheet**.
- **pos**: **number** OPTIONAL
  The position in the **sheets** collection.
- **grid**: **FlexGrid** OPTIONAL
  The **FlexGrid** instance associated with the **Sheet**. If not specified then new **FlexGrid** instance will be created.

継承元 **FlexSheet**
戻り値 **Sheet**

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

addFunction

```
addFunction(name: string, func: Function, description?: string, minParamsCount?: number, maxParamsCount?: number): void
```

Add custom function in **FlexSheet**.

パラメーター

- **name**: **string**
  the name of the custom function.

- **func**: **Function**
  the custom function.
  The function signature looks as follows:

  ```
  function (...params: any[][][]): any;
  ```

  The function takes a variable number of parameters, each parameter corresponds to an expression passed as a function argument. Independently of whether the expression passed as a function argument resolves to a single value or to a cell range, each parameter value is always a two dimensional array of resolved values. The number of rows (first index) and columns (second index) in the array corresponds to the size of the specified cell range. In case where argument is an expression that resolves to a single value, it will be a one-to-one array where its value can be retrieved using the param[0][0] indexer.
  The sample below adds a custom Sum Product function ('customSumProduct') to the FlexSheet:

  ```
  flexSheet.addFunction('customSumProduct', (...params: any[][][]) => {
     let result = 0,
         range1 = params[0],
         range2 = params[1];

     if (range1.length > 0 && range1.length === range2.length && range1[0].length === range2[0].length) {
         for (let i = 0; i < range1.length; i++) {
             for (let j = 0; j < range1[0].length; j++) {
                 result += range1[i][j] * range2[i][j];
             }
         }
     }
     return result;
  }, 'Custom SumProduct Function', 2, 2);
  ```

  After adding this function, it can be used it in sheet cell expressions, like here:

  ```
  =customSumProduct(A1:B5, B1:C5)
  ```

- **description**: **string** OPTIONAL
  the description of the custom function, it will be shown in the function autocompletion of the **FlexSheet**.

- **minParamsCount**: **number** OPTIONAL
  the minimum count of the parameter that the function need.

- **maxParamsCount**: **number** OPTIONAL
  the maximum count of the parameter that the function need. If the count of the parameters in the custom function is arbitrary, the minParamsCount and maxParamsCount should be set to null.

継承元 **FlexSheet**
戻り値 **void**

addUnboundSheet

```
addUnboundSheet(sheetName?: string, rows?: number, cols?: number, pos?: number, grid?: FlexGrid): Sheet
```

Add an unbound **Sheet** to the **FlexSheet**.

パラメーター

- **sheetName**: **string** OPTIONAL
  The name of the Sheet.
- **rows**: **number** OPTIONAL
  The row count of the Sheet.
- **cols**: **number** OPTIONAL
  The column count of the Sheet.
- **pos**: **number** OPTIONAL
  The position in the **sheets** collection.
- **grid**: **FlexGrid** OPTIONAL
  The **FlexGrid** instance associated with the **Sheet**. If not specified then new **FlexGrid** instance will be created.

継承元 **FlexSheet**
戻り値 **Sheet**

applyCellsStyle

```
applyCellsStyle(cellStyle: ICellStyle, cells?: CellRange[], isPreview?: boolean): void
```

Apply the style to a range of cells.

パラメーター

- **cellStyle**: **ICellStyle**
  The **ICellStyle** object to apply.
- **cells**: **CellRange**[] OPTIONAL
  An array of **CellRange** objects to apply the style to. If not specified then style is applied to the currently selected cells.
- **isPreview**: **boolean** OPTIONAL
  Indicates whether the applied style is just for preview.

継承元 **FlexSheet**
戻り値 **void**

applyFunctionToCell

```
applyFunctionToCell(): void
```

Inserts the selected function from the function list to the cell value editor.

継承元 **FlexSheet**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 `Control`
戻り値 **HTMLElement**

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

列をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **c**: **number**
  サイズ変更する列のインデックス。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 `FlexGrid`
戻り値 **void**

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

列の範囲をその内容がちょうど収まるようにサイズ変更します。

測定される行の範囲は常に、現在表示されているすべての行 + 現在表示されていない最大2,000行です。グリッドに大量のデータが含まれている場合には（たとえば、50,000行など）、すべての行を測定すると非常に時間がかかる可能性があるため、すべての行は測定されません。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstColumn**: **number** OPTIONAL
  サイズ変更する最初の列のインデックス（デフォルトは最初の列）。
- **lastColumn**: **number** OPTIONAL
  サイズ変更する最後の列のインデックス（デフォルトは最後の列）。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

行をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **r**: **number**
  サイズ変更する行のインデックス。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の列であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

行の範囲をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックス。
- **lastRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックスす。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の行であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

canEditCell

```
canEditCell(r: number, c: number): void
```

指定されたセルが編集可能かどうかを示す値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

継承元 **FlexGrid**
戻り値 **void**

clear

```
clear(): void
```

Clears the content of the **FlexSheet** control.

継承元 **FlexSheet**
戻り値 **void**

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

すべてのグループ行を指定したレベルに折りたたみます。

パラメーター

- **level**: **number**
  表示する最大のグループレベル。

継承元 **FlexGrid**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

Overrides the base class method to take into account the function list.

継承元 **FlexSheet**
戻り値 **boolean**

STATIC convertNumberToAlpha

```
convertNumberToAlpha(c: number): string
```

Converts the number value to its corresponding alpha value. For instance: 0, 1, 2...to a, b, c...

パラメーター

- **c**: **number**
  The number value need to be converted.

継承元 **FlexSheet**
戻り値 **string**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 `Control`
戻り値 **void**

deleteColumns

```
deleteColumns(index?: number, count?: number): void
```

Deletes columns from the current **Sheet** of the **FlexSheet** control.

パラメーター

- **index**: **number** OPTIONAL
  The starting index of the deleting columns. If not specified then columns will be deleted starting from the first column of the current selection.

- **count**: **number** OPTIONAL
  The numbers of columns to delete. If not specified then one column will be deleted.

継承元 `FlexSheet`
戻り値 **void**

deleteRows

```
deleteRows(index?: number, count?: number): void
```

Deletes rows from the current **Sheet** of the **FlexSheet** control.

パラメーター

- **index**: **number** OPTIONAL
  The starting index of the deleting rows. If not specified then rows will be deleted starting from the first row of the current selection.

- **count**: **number** OPTIONAL
  The numbers of rows to delete. If not specified then one row will be deleted.

継承元 `FlexSheet`
戻り値 **void**

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

継承元 `FlexSheet`
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

evaluate

```
evaluate(formula: string, format?: string, sheet?: Sheet, getPrimitiveResult?: boolean): any
```

Evaluates a formula.

FlexSheet formulas follow the Excel syntax, including a large subset of the functions supported by Excel. A complete list of the functions supported by **FlexSheet** can be found here: **FlexSheet Functions**.

パラメーター

- **formula**: **string**
  The formula to evaluate. The formula may start with an optional equals sign ('=').
- **format**: **string** OPTIONAL
  If specified, defines the .Net format that will be applied to the evaluated value.
- **sheet**: **Sheet** OPTIONAL
  The **Sheet** whose data will be used for evaluation. If not specified then the current sheet is used.
- **getPrimitiveResult**: **boolean** OPTIONAL
  Indicates whether need convert the non-primitive result to primitive type.

継承元 **FlexSheet**
戻り値 **any**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

継承元 **FlexGrid**
戻り値 **boolean**

focus

```
focus(): void
```

グリッド全体をスクロールしてビューに入れることなくグリッドにフォーカスを設定するようにオーバーライドされます。

継承元 **FlexGrid**
戻り値 **void**

freezeAtCursor

```
freezeAtCursor(): void
```

Freeze or unfreeze the columns and rows of the **FlexSheet** control.

継承元 **FlexSheet**
戻り値 **void**

getBuiltInTableStyle

```
getBuiltInTableStyle(styleName: string): TableStyle
```

Get the built-in table style via its name.

パラメーター

- **styleName**: **string**
  The name of the built-in table style.

継承元 **FlexSheet**
戻り値 **TableStyle**

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セル要素の範囲（ビューポート座標単位）を取得します。

このメソッドは、**cells** パネル内のセル（スクロール可能なデータセル）の範囲を返します。その他のパネルにあるセルの範囲を取得するには、該当する**GridPanel** オブジェクトの**getCellBoundingRect** メソッドを使用してください。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

継承元 **FlexGrid**
戻り値 **Rect**

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

グリッドのスクロール可能領域内のセルに格納された値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

継承元 **FlexGrid**
戻り値 **any**

getCellValue

```
getCellValue(rowIndex: number, colIndex: number, formatted?: boolean, sheet?: Sheet): any
```

Gets the evaluated cell value.

Unlike the **getCellData** method that returns a raw data that can be a value or a formula, the **getCellValue** method always returns an evaluated value, that is if the cell contains a formula then it will be evaluated first and the resulting value will be returned.

パラメーター

- **rowIndex**: **number**
  The row index of the cell.
- **colIndex**: **number**
  The column index of the cell.
- **formatted**: **boolean** OPTIONAL
  Indicates whether to return an original or a formatted value of the cell.
- **sheet**: **Sheet** OPTIONAL
  The **Sheet** whose value to evaluate. If not specified then the data from current sheet is used.

継承元 **FlexSheet**
戻り値 **any**

getClipString

```
getClipString(rng?: CellRange): string
```

Gets the content of a **CellRange** as a string suitable for copying to the clipboard.

FlexSheet overrides this method to support multiple rows or columns selection in **FlexSheet**.

Hidden rows and columns are not included in the clip string.

パラメーター

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

継承元 **FlexSheet**
戻り値 **string**

getColumn

```
getColumn(name: string): Column
```

名前または連結に基づいて列を取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **string**
  検索する名前または連結。

継承元 **FlexGrid**
戻り値 **Column**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getMergedRange

```
getMergedRange(panel: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**. This method overrides the getMergedRange method of its parent class FlexGrid

パラメーター

- **panel**: **GridPanel**
  GridPanel that contains the range.
- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

継承元 **FlexSheet**
戻り値 **CellRange**

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

パラメーター

- **r**: **number**
  検査するセルの行インデックス。
- **c**: **number**
  検査するセルの列インデックス。

継承元 **FlexGrid**
戻り値 **SelectedState**

getSelectionFormatState

```
getSelectionFormatState(): IFormatState
```

Gets the **IFormatState** object describing formatting of the selected cells.

継承元 **FlexSheet**
戻り値 **IFormatState**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hideFunctionList

```
hideFunctionList(): void
```

Close the function list.

継承元 **FlexSheet**
戻り値 **void**

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

指定されたポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

次に例を示します。

```
// ユーザーがグリッドをクリックしたときに、ポイントをヒットテストします
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

パラメーター

- **pt**: **any**
  調べる**Point** （ページ座標単位）、MouseEventオブジェクト、またはポイントのX座標。

- **y**: **any** OPTIONAL
  ポイントのY座標（ページ座標単位、最初のパラメーターが数値の場合）。

継承元 **FlexGrid**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

insertColumns

```
insertColumns(index?: number, count?: number): void
```

Inserts columns in the current `Sheet` of the **FlexSheet** control.

パラメーター

- **index**: **number** OPTIONAL
  The position where new columns should be added. If not specified then columns will be added before the left column of the current selection.

- **count**: **number** OPTIONAL
  The numbers of columns to add. If not specified then one column will be added.

継承元 `FlexSheet`
戻り値 **void**

insertRows

```
insertRows(index?: number, count?: number): void
```

Inserts rows in the current **Sheet** of the **FlexSheet** control.

パラメーター

- **index**: **number** OPTIONAL
  The position where new rows should be added. If not specified then rows will be added before the first row of the current selection.

- **count**: **number** OPTIONAL
  The numbers of rows to add. If not specified then one row will be added.

継承元 **FlexSheet**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

指定したCellRangeがこのグリッドの行および列コレクションに対して有効かどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックする範囲。

継承元 **FlexGrid**
戻り値 **boolean**

load

```
load(workbook: any): void
```

Loads the workbook into **FlexSheet**. This method works with JSZip 2.5.

For example:

```
// This sample opens an xlsx file chosen through Open File
// dialog and fills FlexSheet


// HTML
<input type="file"
    id="importFile"
    accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
/>
<div id="flexHost"></>


// JavaScript
var flexSheet = new wijmo.grid.FlexSheet("#flexHost"),
    importFile = document.getElementById('importFile');

importFile.addEventListener('change', function () {
    loadWorkbook();
});

function loadWorkbook() {
    var reader,
        file = importFile.files[0];
    if (file) {
        reader = new FileReader();
        reader.onload = function (e) {
            flexSheet.load(reader.result);
        };
        reader.readAsArrayBuffer(file);
    }
}
```

パラメーター

- **workbook**: **any**
  A workbook instance or a Blob instance or a base-64 string or an ArrayBuffer containing xlsx file content.

継承元 **FlexSheet**
戻り値 **void**

loadAsync

```
loadAsync(workbook: any, onLoaded?: (workbook: wijmo.xlsx.Workbook), onError?: (reason?: any)): void
```

Loads the workbook into **FlexSheet** asynchronously. This method works with JSZip 3.0.

パラメーター

- **workbook**: **any**
  A workbook instance or a Blob instance or a base-64 string or an ArrayBuffer containing xlsx file content.
- **onLoaded**: **(workbook: wijmo.xlsx.Workbook)** OPTIONAL
  This callback provides access to the loaded workbook instance. Since this method is asynchronous, users cannot get the loaded workbook instance immediately. This callback has a single parameter, the loaded workbook instance.
- **onError**: **(reason?: any)** OPTIONAL
  This callback catches errors when loading workbooks. It has a single parameter, the failure reason.

  For example:

```
flexsheet.loadAsync(blob, function (workbook) {

     // user can access the loaded workbook instance in this callback.
     var app = worksheet.application ;
     ...
}, function (reason) {

     // User can catch the failure reason in this callback.
     console.log('The reason of load failure is ' + reason);
});
```

継承元 **FlexSheet**
戻り値 **void**

mergeRange

```
mergeRange(cells?: CellRange, isCopyMergeCell?: boolean): void
```

Merges the selected **CellRange** into one cell.

パラメーター

- **cells**: **CellRange** OPTIONAL
  The **CellRange** to merge.
- **isCopyMergeCell**: **boolean** OPTIONAL
  This parameter indicates that merge operation is done by copy\paste merge cell or not.

継承元 **FlexSheet**
戻り値 **void**

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

autoSizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

autoSizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

autoSizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

autoSizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

beginningEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

cellEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

cellEditEnding イベントを発生させます。

パラメーター

- **e**: **CellEditEndingEventArgs**
  イベントデータを含む**CellEditEndingEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onColumnChanged

```
onColumnChanged(e: RowColumnChangedEventArgs): void
```

Raises the columnChanged event.

パラメーター

- **e**: **RowColumnChangedEventArgs**

継承元 **FlexSheet**
戻り値 **void**

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

copied イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

copying イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

deletedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

deletingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

draggedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

draggedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

draggingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

draggingColumnOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

draggingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRowColumn

```
onDraggingRowColumn(e: DraggingRowColumnEventArgs): void
```

Raises the draggingRowColumn event.

パラメーター

- **e**: **DraggingRowColumnEventArgs**

継承元 **FlexSheet**
戻り値 **void**

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

draggingRowOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDroppingRowColumn

```
onDroppingRowColumn(e?: EventArgs): void
```

Raises the droppingRowColumn event.

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexSheet**
戻り値 **void**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

groupCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

groupCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onItemsSourceChanging

```
onItemsSourceChanging(e: CancelEventArgs): boolean
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLoaded

```
onLoaded(e?: EventArgs): void
```

Raises the loaded event.

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexSheet**
戻り値 **void**

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

loadedRows イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

loadingRows イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

pasted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

pastedCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

pasting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

pastingCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

prepareCellForEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPrepareChangingColumn

```
onPrepareChangingColumn(e: RowColumnChangedEventArgs): void
```

Raises the prepareChangingColumn event.

パラメーター

- **e**: **RowColumnChangedEventArgs**

継承元 **FlexSheet**
戻り値 **void**

onPrepareChangingRow

```
onPrepareChangingRow(e: RowColumnChangedEventArgs): void
```

Raises the prepareChangingRow event.

パラメーター

- **e**: **RowColumnChangedEventArgs**

継承元 **FlexSheet**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

resizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

resizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

resizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

resizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

rowAdded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowChanged

```
onRowChanged(e: RowColumnChangedEventArgs): void
```

Raises the rowChanged event.

パラメーター

- **e**: **RowColumnChangedEventArgs**

継承元 **FlexSheet**
戻り値 **void**

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

rowEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

rowEditEnding イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

rowEditStarted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

rowEditStarting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

scrollPositionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onSelectedSheetChanged

```
onSelectedSheetChanged(e: PropertyChangedEventArgs): void
```

Raises the currentSheetChanged event.

パラメーター

- **e**: **PropertyChangedEventArgs**
  PropertyChangedEventArgs that contains the event data.

継承元 **FlexSheet**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

selectionChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onSheetCleared

```
onSheetCleared(e?: EventArgs): void
```

Raises the sheetCleared event.

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexSheet**
戻り値 **void**

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

sortedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

sortingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUnknownFunction

```
onUnknownFunction(e: UnknownFunctionEventArgs): void
```

Raises the unknownFunction event.

パラメーター

- **e**: **UnknownFunctionEventArgs**

継承元 **FlexSheet**
戻り値 **void**

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

updatedLayout イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

updatingLayout イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

updatingView イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

redo

```
redo(): void
```

Redo the last user action.

継承元 **FlexSheet**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

Overridden to refresh the sheet and the TabHolder.

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

継承元 **FlexSheet**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean**
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

- **recycle**: **boolean** OPTIONAL
  既存の要素を再利用するかどうか。

- **state**: **boolean** OPTIONAL
  既存の要素を保持してその状態を更新するかどうか。

継承元 `FlexGrid`
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この`Control` が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 `Control`
戻り値 **number**

save

```
save(fileName?: string, options?: IFlexSheetXlsxOptions): Workbook
```

Saves **FlexSheet** to xlsx file. This method works with JSZip 2.5.

For example:

```
// This sample exports FlexSheet content to an xlsx file.
// click.


// HTML
<button
    onclick="saveXlsx('FlexSheet.xlsx')">
    Save
</button>


// JavaScript
function saveXlsx(fileName) {

    // Save the flexGrid to xlsx file.
    flexsheet.save(fileName);
}
```

パラメーター

- **fileName**: **string** OPTIONAL
  Name of the file that is generated.

- **options**: **IFlexSheetXlsxOptions** OPTIONAL
  IFlexSheetXlsxOptions object specifying the save options.

継承元 **FlexSheet**
戻り値 **Workbook**

saveAsync

```
saveAsync(fileName?: string, onSaved?: (base64?: string), onError?: (reason?: any), options?: IFlexSheetXlsxOptions): void
```

Saves the **FlexSheet** to xlsx file asynchronously. This method works with JSZip 3.0.

パラメーター

- **fileName**: **string** OPTIONAL
  Name of the file that is generated.

- **onSaved**: **(base64?: string)** OPTIONAL
  This callback provides an approach to get the base-64 string that represents the content of the saved FlexSheet. Since this method is an asynchronous method, user is not able to get the base-64 string immediately. User has to get the base-64 string through this callback. This has a single parameter, the base64 string of the saved flexsheet. It is passed to user.

- **onError**: **(reason?: any)** OPTIONAL
  This callback catches error information when saving. This has a single parameter, the failure reason. The return value is passed to user if he wants to catch the save failure reason.

  For example:

```
flexsheet.saveAsync('', function (base64) {

    // user can access the base64 string in this callback.
    document.getElementById('export').href = 'data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;' + 'base64
,' + base64;
}, function (reason) {

    // User can catch the failure reason in this callback.
    console.log('The reason of save failure is ' + reason);
});
```

- **options**: **IFlexSheetXlsxOptions** OPTIONAL
  `IFlexSheetXlsxOptions` object specifying the save options.

継承元 **`FlexSheet`**
戻り値 **void**

scrollIntoView

```
scrollIntoView(r: number, c: number, refresh?: boolean): boolean
```

指定したセルが画面に入るようにグリッドをスクロールします。

負の行と列のインデックスは無視されるので、以下を呼び出すと、

```
grid.scrollIntoView(200, -1);
```

グリッドは200行目を表示するように垂直にスクロールしますが、水平にスクロールしません。

パラメーター

- **r**: **number**
  画面に入るようにスクロールする行のインデックス

- **c**: **number**
  画面に入るようにスクロールする列のインデックス。

- **refresh**: **boolean** OPTIONAL
  スクロールした新しい位置をすぐ表示するためにグリッドを更新する必要があるかどうかを決定するオプションのパラメータ。

継承元 **`FlexGrid`**
戻り値 **boolean**

select

```
select(rng: any, show?: any): void
```

Selects a cell range and optionally scrolls it into view.

FlexSheet overrides this method to adjust the selection cell range for the merged cells in the **FlexSheet**.

パラメーター

- **rng**: **any**
  The cell range to select.
- **show**: **any** OPTIONAL
  Indicates whether to scroll the new selection into view.

継承元 **FlexSheet**
戻り値 **void**

selectNextFunction

```
selectNextFunction(): void
```

Select next function in the function list.

継承元 **FlexSheet**
戻り値 **void**

selectPreviousFunction

```
selectPreviousFunction(): void
```

Select previous function in the function list.

継承元 **FlexSheet**
戻り値 **void**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

Overrides the setCellData function of the base class.

パラメーター

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **any**
  Index, name, or binding of the column that contains the cell.
- **value**: **any**
  Value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  Whether to change the value automatically to match the column's data type.
- **invalidate**: **boolean** OPTIONAL
  Whether to invalidate the FlexSheet to show the change.

継承元 **FlexSheet**
戻り値 **boolean**

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

Parses a string into rows and columns and applies the content to a given range.

Override the **setClipString** method of **FlexGrid**.

パラメーター

- **text**: **string**
  Tab and newline delimited text to parse into the grid.
- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

継承元 **FlexSheet**
戻り値 **void**

showColumnFilter

```
showColumnFilter(): void
```

Show the filter editor.

継承元 **FlexSheet**
戻り値 **void**

showFunctionList

```
showFunctionList(target: HTMLElement): void
```

Open the function list.

パラメーター

- **target**: **HTMLElement**
  The DOM element that toggle the function list.

継承元 **FlexSheet**
戻り値 **void**

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

指定されたセルの編集を開始します。

FlexGrid の編集は、Excelの編集によく似ています。 ［F2］キーを押すか、セルをダブルクリックすると、 グリッドが**完全編集** モードになります。 このモードでは、ユーザーが［Enter］、［Tab］、または［Esc］キーを押すか、マウスを使用して選択範囲を移動するまで、セルエディタはアクティブのままになります。完全編集モードでは、カーソルキーを押しても、グリッドの編集モードは終了しません。

テキストをセルに直接入力すると、グリッドは**クイック編集**モードになります。 このモードでは、ユーザーがEnter、Tab、Esc、またはいずれかの矢印キーを押すまで、セルエディタはアクティブなままになります。

通常、完全編集モードは既存の値を変更する際に使用します。 クイック編集モードは新しいデータをすばやく入力する際に使用します。

編集中に［F2］キーを押すことで、完全編集モードとクイック編集モードを切り替えることができます。

パラメーター

- **fullEdit**: **boolean** OPTIONAL
  ユーザーがカーソルキーを押したときも編集モードのままにするかどうか。 デフォルトはtrueです。

- **r**: **number** OPTIONAL
  編集する行のインデックス。 デフォルトは現在選択されている行です。

- **c**: **number** OPTIONAL
  編集する列のインデックス。 デフォルトは現在選択されている列です。

- **focus**: **boolean** OPTIONAL
  編集開始時に、エディタにフォーカスを与えるかどうか。 デフォルトはtrueです。

継承元 **FlexGrid**
戻り値 **boolean**

toggleDropDownList

```
toggleDropDownList(): void
```

現在選択されているセルに関連付けられたドロップダウンリストボックスを切り替えます。

このメソッドでは、セルが編集モードに入ったとき、またはユーザが特定のキーを押したときに、ドロップダウンリストを自動的に表示することができます。

たとえば、このコードでは、グリッドが編集モードに入るたびに、グリッドにドロップダウンリストが表示されます。

```
// グリッドが編集モードに入ると、ドロップダウンリストを表示する。
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

このコードでは、ユーザーがスペースバーを押した場合、グリッドが編集モードに入った後で、ドロップダウンリストが表示されます。

```
// ユーザーがスペースバーを押したときにドロップダウンリストを表示する。
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

継承元 **FlexGrid**
戻り値 **void**

undo

```
undo(): void
```

Undo the last user action.

継承元 **FlexSheet**
戻り値 **void**

イベント

autoSizedColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizedRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

beginningEdit

セルが編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnded

セル編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnding

セル編集が終了するときに発生します。

このイベントを使用して検証を実行し、無効な編集を防ぐことができます。 たとえば、次のコードは、ユーザーが文字「a」を含まない値を入力できないようにします。このコードは、編集が適用される前に、編集前と編集後の値を取得する方法を示しています。

```
function cellEditEnding (sender, e) {

  // 編集前と編集後の値を取得します
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // newValに「a」が含まれていない場合は、編集をキャンセルします
  e.cancel = newVal.indexOf('a') < 0;
}
```

cancel パラメータをtrueに設定すると、編集された値が無視されて、グリッドでセルの元の値が維持されます。

また、**stayInEditMode** パラメータをtrueに設定すると、グリッドの編集モードが維持され、編集をコミットする前にユーザーが無効な値を修正できます。

継承元 **FlexGrid**
引数 **CellEditEndingEventArgs**

columnChanged

Occurs after the **FlexSheet** insert\delete columns.

継承元 **FlexSheet**
引数 **RowColumnChangedEventArgs**

copied

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーした後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

copying

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーするときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラでコピー操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletedRow

ユーザーが［Del］キーを押して行を削除した後に発生します （**allowDelete** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletingRow

ユーザーが［Delete］キーを押して選択されている行を削除するときに発生します（**allowDelete** プロパティを参照）。

イベントハンドラで行の削除をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedColumn

ユーザーが列のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedRow

ユーザーが行のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumn

ユーザーが列のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumnOver

ユーザーが列を別の位置にドラッグするときに発生します。

ハンドラは、このイベントをキャンセルして、ユーザーが特定の位置に列をドロップしないようにすることができます。次に例を示します。

```
// ドラッグされている列を記憶します
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// 'sales'列がインデックス0にドラッグされないようにします
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRow

ユーザーが行のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRowColumn

Occurs when dragging the rows or the columns of the **FlexSheet**.

継承元 **FlexSheet**
引数 **DraggingRowColumnEventArgs**

draggingRowOver

ユーザーが行を別の位置にドラッグするときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

droppingRowColumn

Occurs when dropping the rows or the columns of the **FlexSheet**.

継承元 **FlexSheet**
引数 **EventArgs**

formatItem

セルを表す要素が作成されたときに発生します。

このイベントを使用してセルを表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、グループ行のセルから'wj-wrap'クラスを削除します。

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

継承元 **FlexGrid**
引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

groupCollapsedChanged

グループが展開または折りたたまれた後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

groupCollapsedChanging

グループが展開または折りたたまれる直前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

itemsSourceChanged

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

itemsSourceChanging

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

loaded

Occurs after the **FlexSheet** loads the **Workbook** instance

継承元 **FlexSheet**
引数 **EventArgs**

loadedRows

グリッド行がデータソースの項目に連結された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

loadingRows

グリッド行がデータソースの項目に連結される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

pasted

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastedCell

ユーザーがクリップボードの内容をセルに貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pasting

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた時に発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastingCell

ユーザーがクリップボードの内容をセルに貼り付けるときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

prepareCellForEdit

エディタセルが作成されたとき、それがアクティブになる前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

prepareChangingColumn

Occurs before the **FlexSheet** insert\delete columns.

継承元 **FlexSheet**
引数 **RowColumnChangedEventArgs**

prepareChangingRow

Occurs before the **FlexSheet** insert\delete rows.

継承元 **FlexSheet**
引数 **RowColumnChangedEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

resizedColumn

ユーザーが列のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizedRow

ユーザーが行のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingColumn

列がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingRow

行がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowAdded

ユーザーが新規行テンプレートを編集して新しい項目を作成したときに発生します（**allowAddNew** プロパティを参照）。

イベントハンドラで新しい項目の内容をカスタマイズしたり、新しい項目の作成をキャンセルしたりできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowChanged

Occurs after the **FlexSheet** insert\delete rows.

継承元 **FlexSheet**
引数 **RowColumnChangedEventArgs**

rowEditEnded

行編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnding

行編集が終了するとき、変更が確定またはキャンセルされる前に発生します。

このイベントを**rowEditStarted**イベントと組み合わせて使用して、ディープ連結編集を元に戻す操作を実装できます。次に例を示します。

```
// 編集の開始時にディープ連結値を保存します
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.')> -1) { // ディープ連結
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// 編集がキャンセルされたときにディープ連結値を復元します
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // ユーザーによって編集がキャンセルされました
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarted

行が編集モードに入った後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarting

行が編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

scrollPositionChanged

コントロールがスクロールされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

selectedSheetChanged

Occurs when current sheet index changed.

継承元 **FlexSheet**
引数 **PropertyChangedEventArgs**

selectionChanged

選択が変更された後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

selectionChanging

選択が変更される前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sheetCleared

Occurs when the **FlexSheet** is cleared.

継承元 **FlexSheet**
引数 **EventArgs**

sortedColumn

ユーザーが列ヘッダをクリックしてソートを適用した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortingColumn

ユーザーが列ヘッダをクリックしてソートを適用する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

unknownFunction

Occurs when the **FlexSheet** meets the unknown formula.

継承元 **FlexSheet**
引数 **UnknownFunctionEventArgs**

updatedLayout

グリッドで内部レイアウトが更新された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

updatedView

グリッドが現在のビューを構成する要素の作成/更新を完了したときに発生します。

グリッドのビューは以下のような操作に応じて更新されます。

- グリッドまたはそのデータソースの更新
- 行または列の追加、削除、変更
- グリッドのサイズ変更またはスクロール
- 選択の変更

継承元 **FlexGrid**
引数 **EventArgs**

updatingLayout

グリッドで内部レイアウトが更新される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

updatingView

現在のビューを構成する要素の作成/更新をグリッドが開始したときに発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

WjSheet クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.grid.sheet**
基本クラス **Sheet**
表示 ☐継承されたメンバー ☐イベント発生元

Sheet コントロールに対応するAngular 2コンポーネント。

wj-sheetコンポーネントは、 **WjFlexSheet** コンポーネントに含める必要があります。

wj-sheetコンポーネントを使用して、Angular 2アプリケーションに**Sheet**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjSheetコンポーネントは、**Sheet**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- columnCount
- filterSetting
- grid
- initialized
- isInitialized
- itemsSource
- name
- nameChangedNg
- rowCount
- selectionRanges
- tables
- visible
- wjProperty

メソッド

- addTableFromArray
- created
- dispose
- findTable
- getCellStyle
- onNameChanged
- onVisibleChanged

イベント

- nameChanged
- visibleChanged

コンストラクタ

constructor

```
constructor(owner?: FlexSheet, grid?: FlexGrid, sheetName?: string, rows?: number, cols?: number): Sheet
```

Initializes a new instance of the **Sheet** class.

パラメーター

- **owner**: **FlexSheet** OPTIONAL
  The owner @see: FlexSheet control.
- **grid**: **FlexGrid** OPTIONAL
  The associated **FlexGrid** control used to store the sheet data. If not specified then the new **FlexGrid** control will be created.
- **sheetName**: **string** OPTIONAL
  The name of the sheet within the **FlexSheet** control.
- **rows**: **number** OPTIONAL
  The row count for the sheet.
- **cols**: **number** OPTIONAL
  The column count for the sheet.

継承元 **Sheet**
戻り値 **Sheet**

プロパティ

columnCount

Gets or sets the number of columns in the sheet.

継承元 **Sheet**
型 **number**

filterSetting

Gets or sets the filter setting for this sheet.

継承元 **Sheet**
型 **IFilterSetting**

grid

Gets the associated **FlexGrid** control used to store the sheet data.

継承元 **Sheet**
型 **FlexGrid**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemsSource

Gets or sets the array or **`ICollectionView`** for the **`FlexGrid`** instance of the sheet.

継承元 **`Sheet`**
型 **any**

name

Gets or sets the name of the sheet.

継承元 **`Sheet`**
型 **string**

nameChangedNg

プログラムによるアクセスに使用されるWijmo **nameChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **nameChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowCount

Gets or sets the number of rows in the sheet.

継承元 **`Sheet`**
型 **number**

selectionRanges

Gets the selection array.

継承元 **`Sheet`**
型 **`ObservableArray`**

tables

Gets the collection of the **`Table`** objects on this Sheet. It allows to insert/remove **`Table`** on this Sheet via the tables collection.

継承元 **`Sheet`**
型 **`ObservableArray`**

visible

Gets or sets the sheet visibility.

継承元 **`Sheet`**
型 **boolean**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'sheets'です。

型 **string**

メソッド

addTableFromArray

```
addTableFromArray(row: number, column: number, array: any[], properties?: string[], tableName?: string, tableStyle?: TableStyle, options?: ITableOptions, shift?: boolean): Table
```

Add table from an object array.

パラメーター

- **row**: **number**
  The row position of the table.
- **column**: **number**
  The column position of the table.
- **array**: **any**[]
  The object array load to the table.
- **properties**: **string**[] OPTIONAL
  It allows to retrieve only a subset of columns from the object of the array. If it is omitted, the table will load all the keys of the object of the array.
- **tableName**: **string** OPTIONAL
  The name of the table.
- **tableStyle**: **TableStyle** OPTIONAL
  The table style is applied to the table.
- **options**: **ITableOptions** OPTIONAL
  The options **`ITableOptions`** of the table.
- **shift**: **boolean** OPTIONAL
  Indicates whether cells beneath the table should be shifted or not. If not specified cells beneath will be shifted.

継承元 **`Sheet`**
戻り値 **`Table`**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dispose

```
dispose(): void
```

Dispose sheet instance.

継承元 **Sheet**
戻り値 **void**

findTable

```
findTable(rowIndex: number, columnIndex: number): Table
```

Finds the table via the cell location.

パラメーター

- **rowIndex**: **number**
  the row index of the specified cell.
- **columnIndex**: **number**
  the column index of the specified cell.

継承元 **Sheet**
戻り値 **Table**

getCellStyle

```
getCellStyle(rowIndex: number, columnIndex: number): ICellStyle
```

Gets the style of specified cell.

パラメーター

- **rowIndex**: **number**
  the row index of the specified cell.
- **columnIndex**: **number**
  the column index of the specified cell.

継承元 **Sheet**
戻り値 **ICellStyle**

onNameChanged

```
onNameChanged(e: PropertyChangedEventArgs): void
```

Raises the **nameChanged** event.

パラメーター

- **e**: **PropertyChangedEventArgs**

継承元 **Sheet**
戻り値 **void**

onVisibleChanged

```
onVisibleChanged(e: EventArgs): void
```

Raises the **visibleChanged** event.

パラメーター

- **e**: **EventArgs**

継承元 **Sheet**
戻り値 **void**

イベント

nameChanged

Occurs after the sheet name has changed.

継承元 **Sheet**
引数 **PropertyChangedEventArgs**

visibleChanged

Occurs after the visible of sheet has changed.

継承元 **Sheet**
引数 **EventArgs**

wijmo/wijmo.angular2.chart モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart**

wijmo.chartモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.chartは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjChart from 'wijmo/wijmo.angular2.chart';

@Component({
    directives: [wjChart.WjFlexChart, wjChart.WjFlexChartSeries],
    template: `
      <wj-flex-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-flex-chart-series [binding]="'y'"></wj-flex-chart-series>
      </wj-flex-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjFlexChart
- WjFlexChartAxis
- WjFlexChartDataLabel
- WjFlexChartDataPoint
- WjFlexChartLegend
- WjFlexChartLineMarker
- WjFlexChartPlotArea
- WjFlexChartSeries
- WjFlexPie
- WjFlexPieDataLabel

WjFlexChart クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart**
基本クラス **FlexChart**
表示 ☐継承されたメンバー ☐イベント発生元

FlexChart コントロールに対応するAngular 2コンポーネント。

wj-flex-chartコンポーネントを使用して、Angular 2アプリケーションに**FlexChart**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartコンポーネントは、**FlexChart**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-flex-chartコンポーネントには、以下の子コンポーネントを含めます: **WjFlexChartTrendLine** 、 **WjFlexChartMovingAverage** 、 **WjFlexChartYFunctionSeries** 、 **WjFlexChartParametricFunctionSeries** 、 **WjFlexChartWaterfall** 、 **WjFlexChartBoxWhisker** 、 **WjFlexChartErrorBar** 、 **WjFlexChartAnimation** 、 **WjFlexChartAnnotationLayer** 、 **WjFlexChartRangeSelector** 、 **WjFlexChartGestures** 、 **WjFlexChartAxis** 、 **WjFlexChartLegend** 、 **WjFlexChartDataLabel** 、 **WjFlexChartSeries** 、 **WjFlexChartLineMarker** および **WjFlexChartPlotArea**

コンストラクタ

- constructor

プロパティ

- asyncBindings
- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- interpolateNulls
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- lostFocusNg
- options
- palette
- plotAreas
- plotMargin

- renderedNg
- renderingNg
- rightToLeft
- rotated
- selection
- selectionChangedNg
- selectionMode
- series
- seriesVisibilityChangedNg
- stacking
- symbolSize
- tooltip
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FlexChart
```

FlexChart クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト

継承元 **FlexChart**
戻り値 **FlexChart**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

axes

Axis オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

axisX

メインのX軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

- axisY

メインのY軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

- binding

Yの値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

- bindingX

Xデータ値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

- chartType

作成するチャートのタイプを取得または設定します。 The default value for this property is **ChartType.Column**.

継承元 **FlexChart**
型 **ChartType**

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

- dataLabel

ポイントのデータラベルを取得または設定します。

継承元 **FlexChartCore**
型 **DataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **FlexChartCore**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine** 、描画する要素を記述する**HitTestInfo** パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

- itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

- legendToggle

凡例項目をクリックしたときにチャート上の系列の表示/非表示を切り替えるかどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 **FlexChartCore**
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

options

さまざまなチャートオプションを取得または設定します。

以下のオプションがサポートされます。

bubble.maxSize：バブルチャートのシンボルの最大サイズ を指定します。デフォルト値は30ピクセルです。

bubble.minSize：バブルチャートのシンボルの最小サイズ を指定します。デフォルト値は5ピクセルです。

```
chart.options = {
  bubble: { minSize: 5, maxSize: 30 }
}
```

funnel.neckWidth：ファンネルグラフのネックの幅をパーセント値で指定します。 デフォルト値は0.2です。

funnel.neckHeight：ファンネルグラフのネックの高さをパーセント値で指定します。 デフォルト値は0です。

funnel.type：ファンネルグラフのタイプを指定します。これは、'rectangle'または'default'である必要があります。 タイプがrectangleに設定されている場合、neckWidthとneckHeightは機能しません。

```
chart.options = {
  funnel: { neckWidth: 0.3, neckHeight: 0.3, type: 'rectangle' }
}
```

groupWidth：縦棒グラフのグループ幅または 横棒グラフのグループ高さを指定します。グループ幅は、ピクセル単位 または有効なスペースに対するパーセント値で指定できます。デフォルト値は'70%'です。

```
chart.options = {
  groupWidth : 50; // 50ピクセル
}

chart.options = {
  groupWidth : '100%'; // 100%ピクセル
}
```

継承元 **FlexChart**
型 **any**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string[]**

plotAreas

PlotArea オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **PlotAreaCollection**

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

rotated

Xが垂直、Yが水平になるように軸を反転するかどうかを示す値を

取得または設定します。

The default value for this property is **false**.

継承元 **FlexChart**
型 **boolean**

selection

選択されているチャート系列を取得または設定します。

継承元 **FlexChartCore**
型 **SeriesBase**

selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

series

Series オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

seriesVisibilityChangedNg

プログラムによるアクセスに使用されるWijmo **seriesVisibilityChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **seriesVisibilityChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

stacking

系列オブジェクトを積み重ねるかどうかを決定する値と、積み重ねる場合はその方法を取得または設定します。 The default value for this property is **Stacking.None**.

継承元 **FlexChart**
型 **Stacking**

symbolSize

このFlexChart のすべてのSeriesオブジェクトに使用されるシンボルのサイズを取得または設定します。

このプロパティは、各Series オブジェクトのsymbolSizeプロパティによってオーバーライドできます。

The default value for this property is **10** pixels.

継承元 FlexChartCore
型 number

tooltip

チャートのTooltip オブジェクトを取得します。

ツールチップの内容は、次のパラメータを含むテンプレートを使用して生成されます。

- **propertyName**： このポイントによって表されるデータオブジェクトの任意のプロパティ。
- **seriesName**： データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**： データポイントのインデックス。
- **value**： データポイントの**値**（**FlexChart** の場合はy値、**FlexPie** の場合は項目値）。
- **x**： データポイントの**x**値（FlexChartのみ）。
- **y**： データポイントの**y**値（FlexChartのみ）。
- **name**： データポイントの**名前**（**FlexChart** の場合はx値、**FlexPie** の場合は凡例エントリ）。

テンプレートを変更するには、ツールチップのコンテンツプロパティに新しい値を割り当てます。 次に例を示します。

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

チャートのツールチップを無効にできます。それには、テンプレートを空の文字列に設定します。

tooltip プロパティを使用して、次のように、**showDelay** や**hideDelay** などの ツールチップパラメータをカスタマイズすることもできます。

```
chart.tooltip.showDelay = 1000;
```

詳細とオプションについては、**ChartTooltip** プロパティを参照してください。

継承元 FlexChartCore
型 ChartTooltip

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 string

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Point をデータ座標からコントロール座標に変換します。

パラメーター

- **pt**: **any**
  データ座標の**Point** 、またはデータ座標のポイントのX座標。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

seriesVisibilityChanged イベントを発生させます。

パラメーター

- **e**: **SeriesEventArgs**
  イベントデータを含む**SeriesEventArgs** オブジェクト。

継承元 **FlexChartCore**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

pointToData

```
pointToData(pt: any, y?: number): Point
```

Point をコントロール座標からチャートデータ座標に変換します。

パラメーター

- **pt**: **any**
  変換するポイント（コントロール座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**`ImageFormat`** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **`FlexChartBase`**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **`FlexChartBase`**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

seriesVisibilityChanged

系列の表示/非表示が変更されたときに発生します。たとえば、legendToggleプロパティをtrueに設定したり、ユーザーが凡例をクリックしたときです。

継承元 **FlexChartCore**
引数 **SeriesEventArgs**

WjFlexChartAxis クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart**
基本クラス **Axis**
表示 ☐継承されたメンバー ☐イベント発生元

Axis コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-axisコンポーネントは、 **WjFlexChart**、 **WjFlexChartSeries** 、**WjFinancialChart** または、 **WjFinancialChartSeries** コンポーネントの一つに含める必要があります。

wj-flex-chart-axisコンポーネントを使用して、Angular 2アプリケーションに**Axis**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAxisコンポーネントは、**Axis**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- actualMax
- actualMin
- axisLine
- axisType
- binding
- format
- hostElement
- initialized
- isInitialized
- itemFormatter
- itemsSource
- labelAlign
- labelAngle
- labelPadding
- labels
- logBase
- majorGrid
- majorTickMarks
- majorUnit
- max
- min
- minorGrid
- minorTickMarks
- minorUnit
- name
- origin
- overlappingLabels
- plotArea
- position
- rangeChangedNg
- reversed
- title
- wjProperty

メソッド

- convert
- convertBack
- created
- onRangeChanged

イベント

- rangeChanged

コンストラクタ

constructor

```
constructor(position?: Position): Axis
```

Axis クラスの新しいインスタンスを初期化します。

パラメーター

- **position**: **Position** OPTIONAL
  チャート上での軸の位置。

継承元 **Axis**
戻り値 **Axis**

プロパティ

actualMax

実際の軸の最大を取得します。

これは、数値またはDateオブジェクト（時間ベースのデータの場合）を返します。

継承元 **Axis**
型 **any**

actualMin

実際の軸の最小を取得します。

これは、数値またはDateオブジェクト（時間ベースのデータの場合）を返します。

継承元 **Axis**
型 **any**

axisLine

軸線が表示されるかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Axis**
型 **boolean**

axisType

軸タイプを取得します。

継承元 **Axis**
型 **AxisType**

binding

軸ラベルで使用する**itemsSource** プロパティの カンマ区切りのプロパティ名を取得または設定します。

最初の名前は軸の値を指定し、2番目は対応する軸ラベルを表します。デフォルト値は'value,text'です。

継承元 **Axis**
型 **string**

format

軸ラベルに使用される書式文字列を取得または設定します （**Globalize** を参照）。

継承元 **Axis**
型 **string**

hostElement

軸のホスト要素を取得します。

継承元 **Axis**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemFormatter

軸ラベルのitemFormatter関数を取得または設定します。

指定された場合、関数は次の2つのパラメータをとります。

- **render engine**：ラベルの書式設定に使用される**IRenderEngine** オブジェクト。
- **current label**：以下のプロパティを含むオブジェクト。
  - **value**：書式設定する軸ラベルの値。
  - **text**：ラベルに使用するテキスト。
  - **pos**：ラベルがレンダリングされる コントロール座標内の位置。
  - **cls**：ラベルに適用されるCSSクラス。

この関数は、プロパティが変更されるラベルの ラベルパラメータを返します。

次に例を示します。

```
chart.axisY.itemFormatter = function(engine, label) {
    if (label.val > 5){
        engine.textFill = 'red'; // 赤色のテキスト
        label.cls = null; // デフォルトのCSSなし
     }
    return label;
}
```

継承元 **Axis**
型 **Function**

itemsSource

軸ラベルの項目ソースを取得または設定します。

プロパティの名前は、**binding** プロパティによって指定されます。

次に例を示します。

```
// Axis.bindingのデフォルト値は'value,text'です
 chart.axisX.itemsSource = [ { value:1, text:'one' }, { value:2, text:'two' } ];
```

<

継承元 **Axis**
型 **any**

labelAlign

ラベルの配置を取得または設定します。

デフォルトでは、ラベルは中央に配置されます。サポートされている値は、'left'および'right'（x軸の場合）と'top'および'bottom'（y軸の場合）です。

継承元 **Axis**
型 **string**

labelAngle

軸ラベルの回転角度を取得または設定します。

角度は度単位で測定されます。有効な値の範囲は-90～90です。

継承元 **Axis**
型 **number**

labelPadding

ラベルのパディングを取得または設定します。 The default value for this property is **5** pixels.

継承元 **Axis**
型 **number**

labels

軸ラベルを表示すかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Axis**
型 **boolean**

logBase

軸の対数の底を取得または設定します。

底が指定されていない場合、その軸は線形スケールになります。

logBase プロパティを使用すると、原点の周囲に集まっているデータが広がります。これは一部の金融データセットや経済データセットでよく見られます。

継承元 **Axis**
型 **number**

majorGrid

軸のグリッド線が表示されるかどうかを示す値を取得または設定します。

継承元 **Axis**
型 **boolean**

majorTickMarks

軸の目盛りマークの場所を取得または設定します。

継承元 **Axis**
型 **TickMark**

majorUnit

軸ラベル間の単位数を取得または設定します。

軸の値が日付の場合、単位は日数になります。

継承元 **Axis**
型 **number**

max

軸に表示される最大値を取得または設定します。

値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

The default value for this property is **null**, which causes the chart to calculate the maximum value based on the data.

継承元 **Axis**
型 **any**

min

軸に表示される最小値を取得または設定します。

値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

The default value for this property is **null**, which causes the chart to calculate the minimum value based on the data.

継承元 **Axis**
型 **any**

minorGrid

軸の副グリッド線が表示されるかどうかを示す値を取得または設定します。

継承元 **Axis**
型 **boolean**

minorTickMarks

小軸目盛りマークの場所を取得または設定します。

継承元 **Axis**
型 **TickMark**

minorUnit

軸の補助目盛間の単位数を取得または設定します。

軸の値が日付の場合、単位は日数になります。

継承元 **Axis**
型 **number**

name

軸の名前を取得または設定します。

継承元 **Axis**
型 **string**

origin

軸が直交軸と交差する位置の値を取得または設定します。

継承元 **Axis**
型 **number**

overlappingLabels

重なった軸ラベルの処理方法を示す値を取得または設定します。 The default value for this property is **OverlappingLabels.Auto**.

継承元 **Axis**
型 **OverlappingLabels**

plotArea

軸のプロットエリアを取得または設定します。

継承元 **Axis**
型 **PlotArea**

position

プロットエリアに対する軸の位置を取得または設定します。

継承元 **Axis**
型 **Position**

rangeChangedNg

プログラムによるアクセスに使用されるWijmo **rangeChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rangeChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

reversed

軸を反転させる（上から下方向、または右から左方向にする）かどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 **Axis**
型 **boolean**

title

軸の横に表示されるタイトルのテキストを取得または設定します。

継承元 **Axis**
型 **string**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'axes'です。

型 **string**

メソッド

convert

```
convert(val: number, maxValue?: number, minValue?: number): number
```

指定した値をデータ座標からピクセル座標に変換します。

パラメーター

- **val**: **number**
  変換するデータ値。
- **maxValue**: **number** OPTIONAL
  データの最大値（オプション）。
- **minValue**: **number** OPTIONAL
  データの最小値（オプション）。

継承元 **Axis**
戻り値 **number**

convertBack

```
convertBack(val: number): number
```

指定した値をピクセル座標からデータ座標に変換します。

パラメーター

- **val**: **number**
  変換し戻すピクセル座標。

継承元 **Axis**
戻り値 **number**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

rangeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Axis**
戻り値 **void**

イベント

rangeChanged

軸範囲が変更されたときに発生します。

継承元 **Axis**
引数 **EventArgs**

WjFlexChartDataLabel クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart**
基本クラス **DataLabel**
表示 ☐継承されたメンバー ☐イベント発生元

DataLabel コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-data-labelコンポーネントは、 **WjFlexChart** コンポーネントに含める必要があります。

wj-flex-chart-data-labelコンポーネントを使用して、Angular 2アプリケーションに**DataLabel**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartDataLabelコンポーネントは、**DataLabel**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

プロパティ

- border
- connectingLine
- content
- initialized
- isInitialized
- offset
- position
- renderingNg
- wjProperty

メソッド

- created
- onRendering

イベント

- rendering

プロパティ

border

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

継承元 **DataLabelBase**
型 **boolean**

connectingLine

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

継承元 **DataLabelBase**
型 **boolean**

content

データラベルの内容を取得または設定します。

この内容は文字列として指定するほかに、**HitTestInfo** オブジェクトをパラメーターとして受け取る関数として指定することもできます。

ラベルの内容が文字列の場合、以下のパラメーターを含めることができます。

- **seriesName**: データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**: データポイントのインデックス。
- **value**: データポイントの**Value**。
- **x**: データポイントの**x**の値（FlexChartのみ）。
- **y**: データポイントの**y**の値（FlexChartのみ）。
- **name**: データポイントの**Name**。
- **propertyName**: データオブジェクトのプロパティ。

パラメーターは波かっこで囲む必要があります（例: 'x={x}, y={y}'）。

以下の例では、データポイントのyの値をラベルに表示します。

```
// チャートを作成し、データポイントの上に配置したラベルにyのデータを表示します。
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

次の例は、関数を使用してデータラベルの内容を設定する方法を示します。

```
// データラベルの内容を設定します。
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

継承元 **DataLabelBase**
型 **any**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

offset

ラベルからデータポイントまでのオフセットを取得または設定します。

継承元 **DataLabelBase**
型 **number**

position

データラベルの位置を取得または設定します。

継承元 **DataLabel**
型 **LabelPosition**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'dataLabel'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **DataLabelRenderEventArgs**
  ラベルのレンダリングに使用される**DataLabelRenderEventArgs** オブジェクト。

継承元 **DataLabelBase**
戻り値 **void**

イベント

rendering

データラベルをレンダリングする前に発生します。

継承元 **DataLabelBase**
引数 **DataLabelRenderEventArgs**

WjFlexChartDataPoint クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart
基本クラス DataPoint
表示 ☐継承されたメンバー ☐イベント発生元

DataPoint コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-data-pointコンポーネントは、 **WjFlexChartAnnotationText** 、 **WjFlexChartAnnotationEllipse** 、 **WjFlexChartAnnotationRectangle** 、 **WjFlexChartAnnotationLine** 、 **WjFlexChartAnnotationPolygon** 、 **WjFlexChartAnnotationCircle** 、 **WjFlexChartAnnotationSquare** または、 **WjFlexChartAnnotationImage** コンポーネントの一つに含める必要があります。

wj-flex-chart-data-pointコンポーネントを使用して、Angular 2アプリケーションに**DataPoint**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartDataPointコンポーネントは、**DataPoint**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- initialized
- isInitialized
- wjProperty
- x
- y

メソッド

- created

コンストラクタ

constructor

```
constructor(x?: any, y?: any): DataPoint
```

DataPoint クラスの新しいインスタンスを初期化します。

パラメーター

- **x**: **any** OPTIONAL
  新しいDataPointのX 座標。
- **y**: **any** OPTIONAL
  新しいDataPointのY 座標。

継承元 **DataPoint**
戻り値 **DataPoint**

プロパティ

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は"です。

型 **string**

- x

この**DataPoint** のX座標値を取得または設定します。

継承元 **DataPoint**
型 **any**

y

この**DataPoint** のY座標値を取得または設定します。

継承元 **DataPoint**
型 **any**

メソッド

- created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

WjFlexChartLegend クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart**
基本クラス **Legend**
表示 ☐継承されたメンバー ☐イベント発生元

Legend コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-legendコンポーネントは、 **WjFlexChart** 、 **WjFlexPie** 、 **WjFinancialChart** 、 **WjFlexRadar** または、 **WjSunburst** コンポーネントの一つに含める必要があります。

wj-flex-chart-legendコンポーネントを使用して、Angular 2アプリケーションに**Legend**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartLegendコンポーネントは、**Legend**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- initialized
- isInitialized
- position
- title
- titleAlign
- wjProperty

メソッド

- created

コンストラクタ

constructor

```
constructor(chart: FlexChartBase): Legend
```

Legend クラスの新しいインスタンスを初期化します。

パラメーター
- **chart**: **FlexChartBase**
  この**Legend** を所有する**FlexChartBase**。

継承元 **Legend**
戻り値 **Legend**

プロパティ

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- position

凡例を表示するかどうか、表示する場合はプロットエリアに対してどの位置に表示するかを決定する値を取得または設定します。

継承元 **Legend**
型 **Position**

- title

凡例のタイトルを決定する値を取得または設定します。

継承元 **Legend**
型 **string**

- titleAlign

凡例の配置値を決定する値を取得または設定します。 有効な値は、"left" 、"center"または "right" です。

継承元 **Legend**
型 **string**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'legend'です。

型 **string**

メソッド

- created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

WjFlexChartLineMarker クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart
基本クラス LineMarker
表示 ☐継承されたメンバー ☐イベント発生元

LineMarker コントロールに対応するAngular 2コンポーネント。

wj-flex-line-markerコンポーネントは、 **WjFlexChart** または、 **WjFinancialChart** コンポーネントの一つに含める必要があります。

wj-flex-line-markerコンポーネントを使用して、Angular 2アプリケーションに**LineMarker**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartLineMarkerコンポーネントは、**LineMarker**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- alignment
- chart
- content
- dragContent
- dragLines
- dragThreshold
- horizontalPosition
- initialized
- interaction
- isInitialized
- isVisible
- lines
- positionChangedNg
- seriesIndex
- verticalPosition
- wjProperty
- x
- y

メソッド

- created
- onPositionChanged
- remove

イベント

- positionChanged

コンストラクタ

constructor

```
constructor(chart: FlexChartCore, options?): LineMarker
```

LineMarker クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartCore**
  LineMarkerが表示されるチャート。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **LineMarker**
戻り値 **LineMarker**

プロパティ

alignment

LineMarkerの内容の配置を取得または設定します。

デフォルトでは、LineMarkerはターゲットポイントの右下に表示されます。 '|'を使用してAlignmentの値を結合できます。

```
// 配置を左に設定します。
marker.alignment = wijmo.chart.LineMarkerAlignment.Left;

// 配置を左上に設定します。
marker.alignment = wijmo.chart.LineMarkerAlignment.Left | wijmo.chart.LineMarkerAlignment.Top;
```

継承元 **LineMarker**
型 **LineMarkerAlignment**

chart

LineMarkerを所有する**FlexChart** オブジェクトを取得します。

継承元 **LineMarker**
型 **FlexChartCore**

content

LineMarkerのテキストコンテンツをカスタマイズするためのコンテンツ関数を取得または設定します。

継承元 **LineMarker**
型 **Function**

dragContent

インタラクションモードが"Drag"の場合にマーカーの内容をドラッグできるかどうかを示す値を取得または設定します。

継承元 **LineMarker**
型 **boolean**

- dragLines

インタラクションモードが"Drag"の場合に水平線または垂直線をドラッグしたときに両方の線が連動するかどうかを示す値を取得または設定します。

継承元 **LineMarker**
型 **boolean**

- dragThreshold

マーカーをドラッグできる水平線または垂直線からの最大距離を取得または設定します。

継承元 **LineMarker**
型 **number**

- horizontalPosition

プロットエリアに対するLineMarkerの水平位置を取得または設定します。

値の範囲は(0, 1)です。 値がnullまたは未定義で、**interaction** が wijmo.chart.LineMarkerInteraction.Moveまたは wijmo.chart.LineMarkerInteraction.Dragに設定されている場合、マーカーの水平位置はポインタの位置に基づいて自動的に 計算されます。

継承元 **LineMarker**
型 **number**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interaction

LineMarkerのインタラクションモードを取得または設定します。

継承元 **LineMarker**
型 **LineMarkerInteraction**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isVisible

LineMarkerの表示/非表示設定を取得または設定します。

継承元 **LineMarker**
型 **boolean**

- lines

LineMarkerの線の表示/非表示設定を取得または設定します。

継承元 **LineMarker**
型 **LineMarkerLines**

- positionChangedNg

プログラムによるアクセスに使用されるWijmo **positionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **positionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- seriesIndex

このLineMarkerが表示されるチャートの系列のインデックスを取得または設定します。 これは**interaction** プロパティがwijmo.chart.LineMarkerInteraction.Move またはwijmo.chart.LineMarkerInteraction.Dragに設定されているときに有効になります。

継承元 **LineMarker**
型 **number**

- verticalPosition

プロット領域に対するLineMarkerの垂直位置を取得または設定します。

値の範囲は(0, 1)です。 このプロパティの値がnullまたはundefinedで、**interaction** がwijmo.chart.LineMarkerInteraction.Moveまたはwijmo.chart.LineMarkerInteraction.Dragに設定されている場合、LineMarkerの垂直位置はポインタの位置に基づいて自動的に計算されます。

継承元 **LineMarker**
型 **number**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は"です。

型 **string**

- x

現在のxの値をチャートのデータ座標として取得します。

継承元 **LineMarker**
型 **number**

y

現在のyの値をチャートのデータ座標として取得します。

継承元 **LineMarker**
型 **number**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

onPositionChanged

```
onPositionChanged(point: Point): void
```

positionChanged イベントを発生させます。

パラメーター

- **point**: **Point**
  LineMarkerを表示するターゲット位置。

継承元 **LineMarker**
戻り値 **void**

remove

```
remove(): void
```

チャートからLineMarkerを削除します。

継承元 **LineMarker**
戻り値 **void**

イベント

positionChanged

LineMarker の位置が変更された後に発生します。

継承元 **LineMarker**
引数 **Point**

WjFlexChartPlotArea クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart**
基本クラス **PlotArea**
表示 ☐継承されたメンバー ☐イベント発生元

PlotArea コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-plot-areaコンポーネントは、 **WjFlexChart** または、 **WjFinancialChart** コンポーネントの一つに含める必要があります。

wj-flex-chart-plot-areaコンポーネントを使用して、Angular 2アプリケーションに**PlotArea**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartPlotAreaコンポーネントは、**PlotArea**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- column
- height
- initialized
- isInitialized
- name
- row
- style
- width
- wjProperty

メソッド

- created

コンストラクタ

constructor

```
constructor(options?: any): PlotArea
```

PlotArea クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  プロットエリアの初期化オプション。

継承元 **PlotArea**
戻り値 **PlotArea**

プロパティ

column

プロットエリアの列インデックスを取得または設定します。 これにより、チャート上にプロットエリアの水平位置が決定されます。

継承元 **PlotArea**
型 **number**

height

プロットエリアの高さを取得または設定します。

高さは、数値（ピクセル単位）または 書式'{number}*'の文字列（スターサイズ）で指定できます。

継承元 **PlotArea**
型 **any**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

name

プロットエリア名を取得または設定します。

継承元 **PlotArea**
型 **string**

row

プロットエリアの行インデックスを取得または設定します。 これにより、チャート上にプロットエリアの垂直位置が決定されます。

継承元 **PlotArea**
型 **number**

style

プロットエリアのスタイルを取得または設定します。

styleプロパティを使用して、プロットエリアの外観を設定できます。 次に例を示します。 次に例を示します。

```
pa.style = { fill: 'rgba(0,255,0,0.1)' };
```

継承元 **PlotArea**
型 **any**

- width

プロットエリアの幅を取得または設定します。

幅は、数値（ピクセル単位）または書式'{number}*'の文字列（スターサイズ）で指定できます。

継承元 **PlotArea**
型 **any**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'plotAreas'です。

型 **string**

メソッド

- created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

WjFlexChartSeries クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart
基本クラス Series
表示 ☐継承されたメンバー ☐イベント発生元

Series コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-seriesコンポーネントは、 **WjFlexChart** コンポーネントに含める必要があります。

wj-flex-chart-seriesコンポーネントを使用して、Angular 2アプリケーションに**Series**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartSeriesコンポーネントは、**Series**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-chart-series** コンポーネントには、**WjFlexChartAxis** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect

- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): SeriesBase
```

SeriesBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **SeriesBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- chartType

特定の系列のチャートタイプを取得または設定します。これはチャート全体に設定されたチャートタイプをオーバーライドします。 The default value for this property is **null**, which causes the series to use chart type defined by the parent chart.

継承元 **Series**
型 **ChartType**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexPie クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart**
基本クラス **FlexPie**
表示 ☐継承されたメンバー ☐イベント発生元

FlexPie コントロールに対応するAngular 2コンポーネント。

wj-flex-pieコンポーネントを使用して、Angular 2アプリケーションに**FlexPie**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjFlexPieコンポーネントは、**FlexPie**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-flex-pieコンポーネントには、以下の子コンポーネントを含めます: **WjFlexChartAnimation** 、 **WjFlexChartLegend** および **WjFlexPieDataLabel**。

コンストラクタ

- constructor

プロパティ

- binding
- bindingName
- collectionView
- dataLabel
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- innerRadius
- isAnimated
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- lostFocusNg
- offset
- palette
- plotMargin
- renderedNg
- renderingNg
- reversed
- rightToLeft
- selectedIndex
- selectedItemOffset
- selectedItemPosition
- selectionChangedNg

- selectionMode
- startAngle
- tooltip
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FlexPie
```

FlexPie クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  このコントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavascriptオブジェクト。

継承元 **FlexPie**
戻り値 **FlexPie**

プロパティ

binding

チャート値を含むプロパティの名前を取得または設定します。

継承元 **FlexPie**
型 **string**

bindingName

データ項目の名前を含むプロパティの名前を取得または設定します。

継承元 **FlexPie**
型 **string**

collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

dataLabel

ポイントのデータラベルを取得または設定します。

継承元 **FlexPie**
型 **PieDataLabel**

footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- innerRadius

パイの内側半径のサイズを取得または設定します。

内側半径はパイ半径に対する割合として測定されます。

このプロパティのデフォルト値はゼロです（つまり、円グラフになります）。このプロパティをゼロより大きい値に設定すると、円グラフの中央に穴が開きます（これをドーナツグラフと呼びます）。

The default value for this property is **0**

継承元 **FlexPie**
型 **number**

isAnimated

項目が選択されたときにアニメーションを使用するかどうかを示す値を取得または設定します。

selectedItemPosition プロパティおよび**selectionMode** プロパティも参照してください。

The default value for this property is **false**.

継承元 **FlexPie**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

offset

スライスのパイ中心からのオフセットを取得または設定します。

オフセットはパイ半径に対する割合として測定されます。

The default value for this property is **0**.

継承元 **FlexPie**
型 **number**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string**[]

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

reversed

角度を反転（反時計回り）するかどうかを決定する値を取得または設定します。

デフォルト値はfalseです。この場合、角度は時計回りに測定されます。

The default value for this property is **false**.

継承元 **FlexPie**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectedIndex

選択されたスライスのインデックスを取得または設定します。

継承元 **FlexPie**
型 **number**

selectedItemOffset

選択されたスライスのパイ中心からのオフセットを取得または設定します。

オフセットはパイ半径に対する割合として測定されます。

The default value for this property is **0**.

継承元 **FlexPie**
型 **number**

selectedItemPosition

選択されたスライスの位置を取得または設定します。

このプロパティを'None'以外の値に設定すると、スライスを選択したときに円グラフが回転します。

円グラフをクリックしたときにスライスが選択されるようにするには、 **selectionMode** プロパティを'Point'に設定する必要があります。

The default value for this property is **Position.None**.

継承元 **FlexPie**
型 **Position**

selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

startAngle

パイスライスの開始角度（度単位）を取得または設定します。

角度は9時の位置から時計回りに測定されます。

The default value for this property is **0**.

継承元 **FlexPie**
型 **number**

tooltip

チャートの**Tooltip** を取得します。

継承元 **FlexPie**
型 **ChartTooltip**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は''です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexPie**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**ImageFormat** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **FlexChartBase**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **FlexChartBase**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

WjFlexPieDataLabel クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart**
基本クラス **PieDataLabel**
表示 ☐継承されたメンバー ☐イベント発生元

PieDataLabel コントロールに対応するAngular 2コンポーネント。

wj-flex-pie-data-labelコンポーネントは、 **WjFlexPie** コンポーネントに含める必要があります。

wj-flex-pie-data-labelコンポーネントを使用して、Angular 2アプリケーションに**PieDataLabel**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexPieDataLabelコンポーネントは、**PieDataLabel**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

プロパティ

- border
- connectingLine
- content
- initialized
- isInitialized
- offset
- position
- renderingNg
- wjProperty

メソッド

- created
- onRendering

イベント

- rendering

プロパティ

border

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

継承元 **DataLabelBase**
型 **boolean**

connectingLine

データラベルに境界線を表示するかどうかを示す値を取得または設定します。

継承元 **DataLabelBase**
型 **boolean**

content

データラベルの内容を取得または設定します。

この内容は文字列として指定するほかに、**HitTestInfo** オブジェクトをパラメーターとして受け取る関数として指定することもできます。

ラベルの内容が文字列の場合、以下のパラメーターを含めることができます。

- **seriesName**: データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**: データポイントのインデックス。
- **value**: データポイントの**Value**。
- **x**: データポイントの**x**の値（FlexChartのみ）。
- **y**: データポイントの**y**の値（FlexChartのみ）。
- **name**: データポイントの**Name**。
- **propertyName**: データオブジェクトのプロパティ。

パラメーターは波かっこで囲む必要があります（例: 'x={x}, y={y}'）。

以下の例では、データポイントのyの値をラベルに表示します。

```
// チャートを作成し、データポイントの上に配置したラベルにyのデータを表示します。
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

次の例は、関数を使用してデータラベルの内容を設定する方法を示します。

```
// データラベルの内容を設定します。
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

継承元 **DataLabelBase**
型 **any**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

offset

ラベルからデータポイントまでのオフセットを取得または設定します。

継承元 **DataLabelBase**
型 **number**

position

データラベルの位置を取得または設定します。

継承元 **PieDataLabel**
型 **PieLabelPosition**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'dataLabel'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **DataLabelRenderEventArgs**
  ラベルのレンダリングに使用される**DataLabelRenderEventArgs** オブジェクト。

継承元 **DataLabelBase**
戻り値 **void**

イベント

rendering

データラベルをレンダリングする前に発生します。

継承元 **DataLabelBase**
引数 **DataLabelRenderEventArgs**

wijmo/wijmo.angular2.chart.interaction モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.interaction**

wijmo.chart.interactionモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.chart.interactionは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjInteraction from 'wijmo/wijmo.angular2.chart.interaction';
import * as wjChart from 'wijmo/wijmo.angular2.chart';

@Component({
    directives: [wjChart.WjFlexChart, wjInteraction.WjFlexChartRangeSelector, wjChart.WjFlexChartSeries],
    template: `
      <wj-flex-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-flex-chart-range-selector></wj-flex-chart-range-selector>
          <wj-flex-chart-series [binding]="'y'"></wj-flex-chart-series>
      </wj-flex-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjFlexChartGestures
- WjFlexChartRangeSelector

WjFlexChartGestures クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.interaction**
基本クラス **ChartGestures**
表示 ☐継承されたメンバー ☐イベント発生元

ChartGestures コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-gesturesコンポーネントは、 **WjFlexChart** または、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-gesturesコンポーネントを使用して、Angular 2アプリケーションに**ChartGestures**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartGesturesコンポーネントは、**ChartGestures**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- enable
- initialized
- interactiveAxes
- isInitialized
- mouseAction
- posX
- posY
- scaleX
- scaleY
- wjProperty

メソッド

- created
- remove
- reset

コンストラクタ

constructor

```
constructor(chart: FlexChartCore, options?): ChartGestures
```

ChartGestures クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartCore**
  ユーザーがズームまたはパンできる**FlexChart**。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **ChartGestures**
戻り値 **ChartGestures**

プロパティ

- enable

ChartGesturesの有効/無効を取得または設定します。

継承元 `ChartGestures`
型 **boolean**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interactiveAxes

ChartGesturesの操作軸を取得または設定します。

継承元 `ChartGestures`
型 `InteractiveAxes`

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- mouseAction

ChartGesturesのマウス操作を取得または設定します。

継承元 `ChartGestures`
型 `MouseAction`

- posX

軸Xの初期位置を取得または設定します。Scaleが１より小さい場合、この値は軸の初期位置を表します。そうでない場合、Valueは無効です。Valueには０から１までの値を指定する必要があります。

継承元 `ChartGestures`
型 **number**

- posY

軸Yの初期位置を取得または設定します。Scaleが１より小さい場合、この値は軸の初期位置を表します。そうでない場合、Valueは無効です。Valueには０から１までの値を指定する必要があります。

継承元 `ChartGestures`
型 **number**

scaleX

軸Xの初期スケールを取得または設定します。このスケールは０より大きく１以下にする必要があります。 スケールは、最小値から最大値までの範囲のどの部分を表示するかを指定します。スケールが１（デフォルト値）の場合は、軸範囲全体が表示されます。

継承元 **ChartGestures**
型 **number**

scaleY

軸Yの初期スケールを取得または設定します。このスケールは０より大きく１以下にする必要があります。 スケールは、最小値から最大値までの範囲のどの部分を表示するかを指定します。スケールが１（デフォルト値）の場合は、軸範囲全体が表示されます。

継承元 **ChartGestures**
型 **number**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は"です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

remove

```
remove(): void
```

チャートから**ChartGestures** コントロールを削除します。

継承元 **ChartGestures**
戻り値 **void**

reset

```
reset(): void
```

チャートの軸をリセットします。

継承元 **ChartGestures**
戻り値 **void**

WjFlexChartRangeSelector クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.interaction
基本クラス RangeSelector
表示 ☐継承されたメンバー ☐イベント発生元

RangeSelector コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-range-selectorコンポーネントは、 **WjFlexChart** または、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-range-selectorコンポーネントを使用して、Angular 2アプリケーションに**RangeSelector**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartRangeSelectorコンポーネントは、**RangeSelector**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- initialized
- isInitialized
- isVisible
- max
- maxScale
- min
- minScale
- orientation
- rangeChangedNg
- seamless
- wjProperty

メソッド

- created
- onRangeChanged
- remove

イベント

- rangeChanged

コンストラクタ

constructor

```
constructor(chart: FlexChartCore, options?): RangeSelector
```

RangeSelector クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartCore**
  選択された範囲を表示する**FlexChart**。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **RangeSelector**
戻り値 **RangeSelector**

プロパティ

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isVisible

範囲セレクターの表示/非表示設定を取得または設定します。

継承元 **RangeSelector**
型 **boolean**

max

範囲の最大値を取得または設定します。 設定されていない場合、最大値は自動的に計算されます。

継承元 **RangeSelector**
型 **number**

maxScale

チャート範囲全体の割合として選択できるデータの最大量を取得または設定します。 このプロパティは、0と1の間の値に設定する必要があります。

継承元 **RangeSelector**
型 **number**

min

範囲の最小値を取得または設定します。 設定されていない場合、最小値は自動的に計算されます。

継承元 **RangeSelector**
型 **number**

minScale

チャート範囲全体の割合として選択できるデータの最小量を取得または設定します。 このプロパティは、0と1の間の値に設定する必要があります。

継承元 **RangeSelector**
型 **number**

orientation

範囲セレクターの向きを取得または設定します。

継承元 **RangeSelector**
型 **Orientation**

rangeChangedNg

プログラムによるアクセスに使用されるWijmo **rangeChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rangeChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

seamless

最小/最大要素を上下にドラッグして反転できるかどうかを決定する値を取得または設定します。

継承元 **RangeSelector**
型 **boolean**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は"です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

rangeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **RangeSelector**
戻り値 **void**

remove

```
remove(): void
```

チャートから**RangeSelector** コントロールを削除します。

継承元 **RangeSelector**
戻り値 **void**

イベント

rangeChanged

範囲が変更された後に発生します。

継承元 **RangeSelector**
引数 **EventArgs**

wijmo/wijmo.angular2.chart.animation モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.animation**

wijmo.chart.animationモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.chart.animationは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjAnimation from 'wijmo/wijmo.angular2.chart.animation';
import * as wjChart from 'wijmo/wijmo.angular2.chart';

@Component({
    directives: [wjChart.WjFlexChart, wjAnimation.WjFlexChartAnimation, wjChart.WjFlexChartSeries],
    template: `
      <wj-flex-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-flex-chart-animation [animationMode]="'Point'"></wj-flex-chart-animation>
          <wj-flex-chart-series [binding]="'y'"></wj-flex-chart-series>
      </wj-flex-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjFlexChartAnimation

WjFlexChartAnimation クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.animation**
基本クラス **ChartAnimation**
表示 ☐継承されたメンバー ☐イベント発生元

ChartAnimation コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-animationコンポーネントは、 **WjFlexChart** 、 **WjFlexPie** 、 **WjFinancialChart** または、 **WjFlexRadar** コンポーネントの一つ に含める必要があります。

wj-flex-chart-animationコンポーネントを使用して、Angular 2アプリケーションに**ChartAnimation**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnimationコンポーネントは、**ChartAnimation**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- animationMode
- axisAnimation
- duration
- easing
- initialized
- isInitialized
- wjProperty

メソッド

- animate
- created

コンストラクタ

constructor

```
constructor(chart: FlexChartBase, options?: any): ChartAnimation
```

ChartAnimation クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartBase**
  ChartAnimation の添付先のチャート。

- **options**: **any** OPTIONAL
  ChartAnimation の初期化データを含むJavaScriptオブジェクト。

継承元 **ChartAnimation**
戻り値 **ChartAnimation**

プロパティ

- animationMode

プロットポイントを一度に 1 つずつアニメーションするか、系列ごとにアニメーションするか、一度にすべてをアニメーションするかを取得または設定します。 アニメーション全体は時間内に完了します。

継承元 **ChartAnimation**
型 **AnimationMode**

- axisAnimation

アニメーションが軸に適用されるかどうかを示す値を取得または設定します。

継承元 **ChartAnimation**
型 **boolean**

- duration

アニメーション全体の長さ（ミリ秒）を取得または設定します。

継承元 **ChartAnimation**
型 **number**

- easing

アニメーションに適用されるイージング関数を取得または設定します。

継承元 **ChartAnimation**
型 **Easing**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は"です。

型 **string**

メソッド

animate

```
animate(): void
```

アニメーションを実行します。

継承元 **ChartAnimation**
戻り値 **void**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

wijmo/wijmo.angular2.chart.analytics モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.chart.analytics**

wijmo.chart.analyticsモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.chart.analyticsは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjAnalitics from 'wijmo/wijmo.angular2.chart.analytics';
```

クラス

- WjFlexChartBoxWhisker
- WjFlexChartErrorBar
- WjFlexChartMovingAverage
- WjFlexChartParametricFunctionSeries
- WjFlexChartTrendLine
- WjFlexChartWaterfall
- WjFlexChartYFunctionSeries

WjFlexChartBoxWhisker クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.analytics
基本クラス BoxWhisker
表示 ☐継承されたメンバー ☐イベント発生元

BoxWhisker コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-box-whiskerコンポーネントは、

WjFlexChart または**WjFinancialChart** コンポーネントのいずれかに含める必要があります。

wj-flex-chart-box-whiskerコンポーネントを使用して、Angular 2アプリケーションに**BoxWhisker**コントロールを追加できます。 Angular 2マークアップの構文については、 「Angular 2マークアップ」を参照してください。

WjFlexChartBoxWhiskerコンポーネントは、**BoxWhisker**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- gapWidth
- groupWidth
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- meanLineStyle
- meanMarkerStyle
- name
- quartileCalculation
- renderedNg
- renderingNg
- showInnerPoints
- showMeanLine
- showMeanMarker
- showOutliers
- style
- symbolMarker
- symbolSize

- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): BoxWhisker
```

BoxWhisker クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **BoxWhisker**
戻り値 **BoxWhisker**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

gapWidth

グループ間のギャップ幅をパーセント値で指定する値を取得または設定します。

このプロパティのデフォルト値は0.1です。最小値は0で、最大値は1です。

継承元 `BoxWhisker`
型 **number**

groupWidth

グループ幅をパーセント値で指定する値を取得または設定します。

このプロパティのデフォルト値は0.8です。最小値は0で、最大値は1です。

継承元 `BoxWhisker`
型 **number**

hostElement

系列のホスト要素を取得します。

継承元 `SeriesBase`
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 `SeriesBase`
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemsSource

系列データを含む配列または`ICollectionView` オブジェクトを取得または設定します。

継承元 `SeriesBase`
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

meanLineStyle

平均線のスタイルを指定する値を取得または設定します。

継承元 **BoxWhisker**
型 **any**

meanMarkerStyle

平均マーカーのスタイルを指定する値を取得または設定します。

継承元 **BoxWhisker**
型 **any**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

quartileCalculation

四分位数計算方法を指定する値を取得または設定します。

継承元 **BoxWhisker**
型 **QuartileCalculation**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

showInnerPoints

系列の各ポイントの内側データポイントを表示するかどうかを決定する値を取得または設定します。

継承元 **BoxWhisker**
型 **boolean**

showMeanLine

平均線を表示するかどうかを指定する値を取得または設定します。

継承元 **BoxWhisker**
型 **boolean**

showMeanMarker

平均マーカーを表示するかどうかを指定する値を取得または設定します。

継承元 **BoxWhisker**
型 **boolean**

showOutliers

異常値を表示するかどうかを指定する値を取得または設定します。

異常値は、第1と第3の四分位間の範囲外にある内側ポイントです。

継承元 **BoxWhisker**
型 **boolean**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

WjFlexChartErrorBar クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.analytics**
基本クラス **ErrorBar**
表示 ☐継承されたメンバー ☐イベント発生元

ErrorBar コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-error-barコンポーネントは、 **WjFlexChart** コンポーネントに含める必要があります。

wj-flex-chart-error-barコンポーネントを使用して、Angular 2アプリケーションに**ErrorBar**コントロールを追加できます。 Angular 2マークアップの構文については、 「Angular 2マークアップ」を参照してください。

WjFlexChartErrorBarコンポーネントは、**ErrorBar**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- direction
- endStyle
- errorAmount
- errorBarStyle
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- value
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): ErrorBar
```

ErrorBar クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **ErrorBar**
戻り値 **ErrorBar**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- chartType

特定の系列のチャートタイプを取得または設定します。これはチャート全体に設定されたチャートタイプをオーバーライドします。 The default value for this property is **null**, which causes the series to use chart type defined by the parent chart.

継承元 **Series**
型 **ChartType**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

direction

誤差範囲バーの方向を指定する値を取得または設定します。

継承元 **ErrorBar**
型 **ErrorBarDirection**

endStyle

誤差範囲バーの終点スタイルを指定する値を取得または設定します。

継承元 **ErrorBar**
型 **ErrorBarEndStyle**

errorAmount

value プロパティの意味を指定する値を取得または設定します。

継承元 **ErrorBar**
型 **ErrorAmount**

errorBarStyle

誤差範囲バーのレンダリングに使用されるスタイルを取得または設定します。

継承元 **ErrorBar**
型 **any**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- value

系列の誤差値を指定する値を取得または設定します。

このプロパティは、**errorAmount** プロパティと組み合わせて使用されます。

継承元 **ErrorBar**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **ErrorBar**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartMovingAverage クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.analytics
基本クラス MovingAverage
表示 ☐継承されたメンバー ☐イベント発生元

MovingAverage コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-moving-averageコンポーネントは、 **WjFlexChart** または、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-moving-averageコンポーネントを使用して、Angular 2アプリケーションに**MovingAverage**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartMovingAverageコンポーネントは、**MovingAverage**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- type
- visibility
- wjProperty

メソッド

- approximate
- created

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): MovingAverage
```

MovingAverage クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **MovingAverage**
戻り値 **MovingAverage**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

移動平均系列の期間を取得または設定します。 これは、1より大きな整数値に設定する必要があります。

継承元 **MovingAverage**
型 **number**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

継承元 **TrendLineBase**
型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

type

移動平均系列のタイプを取得または設定します。

継承元 **MovingAverage**
型 **MovingAverageType**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

approximate

```
approximate(x: number): number
```

指定されたx値から近似y値を取得します。

パラメーター

- **x**: **number**
  y値の計算に使用されるx値。

継承元 **TrendLineBase**
戻り値 **number**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartParametricFunctionSeries クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.analytics
基本クラス ParametricFunctionSeries
表示 ☐継承されたメンバー ☐イベント発生元

ParametricFunctionSeries コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-parametric-function-seriesコンポーネントは、 **WjFlexChart** または、 **WjFinancialChart** コンポーネントに含まれる必要があります。

wj-flex-chart-parametric-function-seriesコンポーネントを使用して、Angular 2アプリケーションに**ParametricFunctionSeries**コントロールを追加します。 Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartParametricFunctionSeriesコンポーネントは、**ParametricFunctionSeries**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- max
- min
- name
- renderedNg
- renderingNg
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty
- xFunc
- yFunc

メソッド

- approximate
- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): ParametricFunctionSeries
```

ParametricFunctionSeries クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **ParametricFunctionSeries**
戻り値 **ParametricFunctionSeries**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- max

関数を計算するためのパラメータの最大値を取得または設定します。

継承元 **FunctionSeries**
型 **number**

- min

関数を計算するためのパラメータの最小値を取得または設定します。

継承元 **FunctionSeries**
型 **number**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

継承元 **TrendLineBase**
型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

このSeries をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 SeriesBase
型 number

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 SeriesBase
型 any

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 SeriesBase
型 SeriesVisibility

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 string

- xFunc

x値の計算に使用される関数を取得または設定します。

継承元 ParametricFunctionSeries
型 Function

- yFunc

y値の計算に使用される関数を取得または設定します。

継承元 ParametricFunctionSeries
型 Function

メソッド

approximate

```
approximate(value: number): void
```

指定された値から近似xおよびyを取得します。

パラメーター

- **value**: **number**
  計算する値。

継承元 **ParametricFunctionSeries**
戻り値 **void**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartTrendLine クラス

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.chart.analytics**
基本クラス **TrendLine**
表示 ☐継承されたメンバー ☐イベント発生元

TrendLine コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-trend-lineコンポーネントは、 **WjFlexChart** または、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-trend-lineコンポーネントを使用して、Angular 2アプリケーションに**TrendLine**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartTrendLineコンポーネントは、**TrendLine**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- coefficients
- collectionView
- cssClass
- fitType
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- order
- renderedNg
- renderingNg
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- approximate

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getEquation
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): TrendLine
```

TrendLine クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **TrendLine**
戻り値 **TrendLine**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- coefficients

計算式の係数を取得します。

継承元 **TrendLine**
型 **number**[]

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

fitType

TrendLine のフィッティングタイプを取得または設定します。

継承元 **TrendLine**
型 **TrendLineFitType**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 `SeriesBase`
型 **string**

- order

多項式またはフーリエ方程式の項の数を取得または設定します。

この値は、1より大きい整数に設定してください。 これは、fitTypeがwijmo.chart.analytics.TrendLineFitType.Polynomialまたは wijmo.chart.analytics.TrendLineFitType.Fourier に設定されたときに適用されます。

継承元 `TrendLine`
型 **number**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

継承元 `TrendLineBase`
型 **number**

- style

系列のスタイルを取得または設定します。

継承元 `SeriesBase`
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

approximate

```
approximate(x: number): number
```

指定されたx値から近似y値を取得します。

パラメーター

- **x**: **number**
  y値の計算に使用されるx値。

継承元 **TrendLine**
戻り値 **number**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getEquation

```
getEquation(fmt?: Function): void
```

係数の書式設定された式文字列を取得します。

パラメーター

- **fmt**: **Function** OPTIONAL
  係数を文字列に変換するために使用される書式設定関数。 このパラメータはオプションです。

継承元 **TrendLine**
戻り値 **void**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

WjFlexChartWaterfall クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.analytics**
基本クラス **Waterfall**
表示 ☐継承されたメンバー ☐イベント発生元

Waterfall コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-waterfallコンポーネントは、 **WjFlexChart** または、 **WjFinancialChart** コンポーネントに含まれる必要があります。

wj-flex-chart-waterfallコンポーネントを使用して、Angular 2アプリケーションに**Waterfall**コントロールを追加します。 Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartWaterfallコンポーネントは、**Waterfall**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- connectorLines
- cssClass
- hostElement
- initialized
- intermediateTotalLabels
- intermediateTotalPositions
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- relativeData
- renderedNg
- renderingNg
- showIntermediateTotal
- showTotal
- start
- startLabel
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- totalLabel

- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Waterfall
```

Waterfall クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Waterfall**
戻り値 **Waterfall**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- connectorLines

接続線を表示するかどうかを決定する値を取得または設定します。

継承元 **Waterfall**
型 **boolean**

cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

intermediateTotalLabels

小計バーのラベルを含むプロパティの名前を取得または設定します。この名前は配列または文字列である必要があります。

このプロパティは、**showIntermediateTotal** プロパティおよび**intermediateTotalPositions** プロパティと組み合わせて使用されます。

継承元 **Waterfall**
型 **any**

intermediateTotalPositions

小計バーの位置のインデックスを含むプロパティの値を取得または設定します。

このプロパティは、**showIntermediateTotal** プロパティおよび**intermediateTotalLabels** プロパティと組み合わせて使用されます。

継承元 **Waterfall**
型 **number[]**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- relativeData

指定されたデータが相対値（差異）であるかどうかを決定する値を取得または設定します。

継承元 **Waterfall**
型 **boolean**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- showIntermediateTotal

小計バーを表示するかどうかを決定する値を取得または設定します。

このプロパティは、**intermediateTotalPositions** プロパティおよび**intermediateTotalLabels** プロパティと組み合わせて使用されます。

継承元 **Waterfall**
型 **boolean**

showTotal

チャートの末尾に合計バーを表示するかどうかを決定する値を取得または設定します。

継承元 **Waterfall**
型 **boolean**

start

開始バーの値を決定する値を取得または設定します。 開始値がnullの場合、開始バーは表示されません。

継承元 **Waterfall**
型 **number**

startLabel

開始バーのラベルを取得または設定します。

継承元 **Waterfall**
型 **string**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

styles

ウォータフォールのスタイルを取得または設定します。

以下のスタイルがサポートされています。

1. **start**: 開始バーのスタイルを指定します。
2. **total**: 合計バーのスタイルを指定します。
3. **intermediateTotal**: 小計バーのスタイルを指定します。
4. **falling**: 減少バーのスタイルを指定します。
5. **rising**: 増加バーのスタイルを指定します。
6. **connectorLines**: 接続線のスタイルを指定します。

```
waterfall.styles = {
  start: { fill: 'blue', stroke: 'blue' },
  total: { fill: 'yellow', stroke: 'yellow' },
  falling: { fill: 'red', stroke: 'red' },
  rising: { fill: 'green', stroke: 'green' },
  connectorLines: { stroke: 'blue', 'stroke-dasharray': '10, 10' }
}
```

継承元 **Waterfall**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- totalLabel

合計バーのラベルを取得または設定します。

継承元 **Waterfall**
型 **string**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartYFunctionSeries クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.analytics
基本クラス YFunctionSeries
表示 ☐継承されたメンバー ☐イベント発生元

YFunctionSeries コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-y-function-seriesコンポーネントは、 **WjFlexChart** または **WjFinancialChart** コンポーネントに含まれる必要があります。

wj-flex-chart-y-function-seriesコンポーネントを使用して、Angular 2アプリケーションに**YFunctionSeries**コントロールを追加します。 Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartYFunctionSeriesコンポーネントは、**YFunctionSeries**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- func
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- max
- min
- name
- renderedNg
- renderingNg
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- approximate

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): YFunctionSeries
```

YFunctionSeries クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **YFunctionSeries**
戻り値 **YFunctionSeries**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- func

Y値の計算に使用される関数を取得または設定します。

継承元 **YFunctionSeries**
型 **Function**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- max

関数を計算するためのパラメータの最大値を取得または設定します。

継承元 **FunctionSeries**
型 **number**

min

関数を計算するためのパラメータの最小値を取得または設定します。

継承元 **FunctionSeries**
型 **number**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

sampleCount

関数計算のためのサンプル数を取得または設定します。 このプロパティは、MovingAverageに適用されません。

継承元 **TrendLineBase**
型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

このSeriesをレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 SeriesBase
型 number

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 SeriesBase
型 any

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 SeriesBase
型 SeriesVisibility

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 string

メソッド

approximate

```
approximate(x: number): number
```

指定されたx値から近似y値を取得します。

パラメーター

- **x**: **number**
  y値の計算に使用されるx値。

継承元 YFunctionSeries
戻り値 number

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

wijmo/wijmo.angular2.chart.annotation モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.annotation**

wijmo.chart.annotationモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.chart.annotationは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjAnnotation from 'wijmo/wijmo.angular2.chart.annotation';
import * as wjChart from 'wijmo/wijmo.angular2.chart';

@Component({
    directives: [wjChart.WjFlexChart, wjAnnotation.WjFlexChartAnnotationLayer,
           wjAnnotation.WjFlexChartAnnotationCircle, wjChart.WjFlexChartSeries],
    template: `
      <wj-flex-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-flex-chart-series [binding]="'y'"></wj-flex-chart-series>
          <wj-flex-chart-annotation-layer>
              <wj-flex-chart-annotation-circle [radius]="40" [point]="{x: 250, y: 150}"></wj-flex-chart-annotation-circle>
          </wj-flex-chart-annotation-layer>
      </wj-flex-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjFlexChartAnnotationCircle
- WjFlexChartAnnotationEllipse
- WjFlexChartAnnotationImage
- WjFlexChartAnnotationLayer
- WjFlexChartAnnotationLine
- WjFlexChartAnnotationPolygon
- WjFlexChartAnnotationRectangle
- WjFlexChartAnnotationSquare
- WjFlexChartAnnotationText

WjFlexChartAnnotationCircle クラス

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.chart.annotation**
基本クラス **Circle**
表示 ☐継承されたメンバー ☐イベント発生元

Circle コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-annotation-circleコンポーネントは、 **WjFlexChartAnnotationLayer** コンポーネントに含める必要があります。

wj-flex-chart-annotation-circleコンポーネントを使用して、Angular 2アプリケーションに**Circle**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnnotationCircleコンポーネントは、**Circle**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-chart-annotation-circle** コンポーネントには、 **WjFlexChartDataPoint** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- radius
- seriesIndex
- style
- tooltip
- wjProperty

メソッド

- created
- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Circle
```

Circle 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Circle**
戻り値 **Circle**

プロパティ

attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

- radius

Circle 注釈の半径を取得または設定します。

継承元 **Circle**
型 **number**

- seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

- tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'items'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

WjFlexChartAnnotationEllipse クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.annotation**
基本クラス **Ellipse**
表示 ☐継承されたメンバー ☐イベント発生元

Ellipse コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-annotation-ellipseコンポーネントは、 **WjFlexChartAnnotationLayer** コンポーネントに含める必要があります。

wj-flex-chart-annotation-ellipseコンポーネントを使用して、Angular 2アプリケーションに**Ellipse**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnnotationEllipseコンポーネントは、**Ellipse**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-chart-annotation-ellipse** コンポーネントには、 **WjFlexChartDataPoint** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- height
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width
- wjProperty

メソッド

- created
- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Ellipse
```

Ellipse 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Ellipse**
戻り値 **Ellipse**

プロパティ

attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

height

Ellipse 注釈の高さを取得または設定します。

継承元 **Ellipse**
型 **number**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

width

Ellipse 注釈の幅を取得または設定します。

継承元 **Ellipse**
型 **number**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'items'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

WjFlexChartAnnotationImage クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.annotation
基本クラス Image
表示 ☐ 継承されたメンバー ☐ イベント発生元

Image コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-annotation-imageコンポーネントは、 **WjFlexChartAnnotationLayer** コンポーネントに含める必要があります。

wj-flex-chart-annotation-imageコンポーネントを使用して、Angular 2アプリケーションに**Image**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnnotationImageコンポーネントは、**Image**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-chart-annotation-image** コンポーネントには、 **WjFlexChartDataPoint** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- height
- href
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width
- wjProperty

メソッド

- created
- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Image
```

Image 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Image**
戻り値 **Image**

プロパティ

attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

height

Image 注釈の高さを取得または設定します。

継承元 **Image**
型 **number**

href

Image 注釈のhrefを取得または設定します。

継承元 **Image**
型 **string**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

- name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

- seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

width

Image 注釈の幅を取得または設定します。

継承元 **Image**
型 **number**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'items'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

WjFlexChartAnnotationLayer クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.annotation
基本クラス AnnotationLayer
表示 ☐継承されたメンバー ☐イベント発生元

AnnotationLayer コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-annotation-layerコンポーネントは、 **WjFlexChart** または、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-annotation-layerコンポーネントを使用して、Angular 2アプリケーションに**AnnotationLayer**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnnotationLayerコンポーネントは、**AnnotationLayer**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-flex-chart-annotation-layerコンポーネントには、 以下の子コンポーネントを含めます: **WjFlexChartAnnotationText** 、 **WjFlexChartAnnotationEllipse** 、 **WjFlexChartAnnotationRectangle** 、 **WjFlexChartAnnotationLine** 、 **WjFlexChartAnnotationPolygon** 、 **WjFlexChartAnnotationCircle** 、 **WjFlexChartAnnotationSquare** および **WjFlexChartAnnotationImage**。

コンストラクタ

constructor

```
constructor(chart: FlexChartCore, options?): AnnotationLayer
```

AnnotationLayer クラスの新しいインスタンスを初期化します。

パラメーター

- **chart**: **FlexChartCore**
  AnnotationLayer の添付先のチャート。
- **options**: OPTIONAL
  AnnotationLayer の初期化データを含むJavaScriptオブジェクト。

継承元 **AnnotationLayer**
戻り値 **AnnotationLayer**

プロパティ

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- items

AnnotationLayer 内の注釈要素のコレクションを取得します。

継承元 **AnnotationLayer**
型 **ObservableArray**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は"です。

型 **string**

メソッド

- created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

- getItem

```
getItem(name: string): AnnotationBase
```

AnnotationLayer 内の注釈要素を名前に基づいて取得します。

パラメーター

- **name**: **string**
  注釈の名前。

継承元 **AnnotationLayer**
戻り値 **AnnotationBase**

getItems

```
getItems(name: string): Array
```

AnnotationLayer 内の複数の注釈要素を名前に基づいて取得します。

パラメーター

- **name**: **string**
  注釈の名前。

継承元 **AnnotationLayer**
戻り値 **Array**

WjFlexChartAnnotationLine クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.annotation
基本クラス Line
表示 ☐継承されたメンバー ☐イベント発生元

Line コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-annotation-lineコンポーネントは、 **WjFlexChartAnnotationLayer** コンポーネントに含める必要があります。

wj-flex-chart-annotation-lineコンポーネントを使用して、Angular 2アプリケーションに**Line**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnnotationLineコンポーネントは、**Line**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-chart-annotation-line** コンポーネントには、 **WjFlexChartDataPoint** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- end
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- start
- style
- tooltip
- wjProperty

メソッド

- created
- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Line
```

Line 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Line**
戻り値 **Line**

プロパティ

attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

end

Line注釈の終了点を取得または設定します。

継承元 **Line**
型 **DataPoint**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

start

Line 注釈の開始点を取得または設定します。

継承元 **Line**
型 **DataPoint**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'items'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

WjFlexChartAnnotationPolygon クラス

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.chart.annotation**
基本クラス **Polygon**
表示 ☐継承されたメンバー ☐イベント発生元

Polygon コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-annotation-polygonコンポーネントは、 **WjFlexChartAnnotationLayer** コンポーネントに含める必要があります。

wj-flex-chart-annotation-polygonコンポーネントを使用して、Angular 2アプリケーションに**Polygon**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnnotationPolygonコンポーネントは、**Polygon**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-chart-annotation-polygon** コンポーネントには、 **WjFlexChartDataPoint** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- points
- position
- seriesIndex
- style
- tooltip
- wjProperty

メソッド

- created
- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Polygon
```

Polygon 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Polygon**
戻り値 **Polygon**

プロパティ

attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

- offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

- point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

- pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- points

Polygon 注釈のポイントのコレクションを取得します。

継承元 **Polygon**
型 **ObservableArray**

- position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

- seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

- style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

- tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'items'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

WjFlexChartAnnotationRectangle クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.annotation
基本クラス Rectangle
表示 ☐継承されたメンバー ☐イベント発生元

Rectangle コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-annotation-rectangleコンポーネントは、 **WjFlexChartAnnotationLayer** コンポーネントに含める必要があります。

wj-flex-chart-annotation-rectangleコンポーネントを使用して、Angular 2アプリケーションに**Rectangle**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnnotationRectangleコンポーネントは、**Rectangle**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-chart-annotation-rectangle** コンポーネントには、 **WjFlexChartDataPoint** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- height
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width
- wjProperty

メソッド

- created
- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Rectangle
```

Rectangle 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Rectangle**
戻り値 **Rectangle**

プロパティ

attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

height

Rectangle 注釈の高さを取得または設定します。

継承元 **Rectangle**
型 **number**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

width

Rectangle 注釈の幅を取得または設定します。

継承元 **Rectangle**
型 **number**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'items'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

WjFlexChartAnnotationSquare クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.annotation**
基本クラス **Square**
表示 ☐継承されたメンバー ☐イベント発生元

Square コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-annotation-squareコンポーネントは、 **WjFlexChartAnnotationLayer** コンポーネントに含める必要があります。

wj-flex-chart-annotation-squareコンポーネントを使用して、Angular 2アプリケーションに**Square**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnnotationSquareコンポーネントは、**Square**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-chart-annotation-square** コンポーネントには、 **WjFlexChartDataPoint** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- attachment
- content
- initialized
- isInitialized
- isVisible
- length
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- wjProperty

メソッド

- created
- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Square
```

Square 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Square**
戻り値 **Square**

プロパティ

attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

content

注釈のテキストを取得または設定します。

継承元 **Shape**
型 **string**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

length

Square 注釈の長さを取得または設定します。

継承元 **Square**
型 **number**

name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'items'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

WjFlexChartAnnotationText クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.annotation
基本クラス Text
表示 ☐継承されたメンバー ☐イベント発生元

Text コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-annotation-textコンポーネントは、 **WjFlexChartAnnotationLayer** コンポーネントに含める必要があります。

wj-flex-chart-annotation-textコンポーネントを使用して、Angular 2アプリケーションに**Text**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartAnnotationTextコンポーネントは、**Text**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-flex-chart-annotation-text** コンポーネントには、 **WjFlexChartDataPoint** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- attachment
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- text
- tooltip
- wjProperty

メソッド

- created
- destroy
- render

コンストラクタ

constructor

```
constructor(options?: any): Text
```

Text 注釈クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Text**
戻り値 **Text**

プロパティ

attachment

注釈の添付方法を取得または設定します。

継承元 **AnnotationBase**
型 **AnnotationAttachment**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isVisible

注釈の表示/非表示を取得または設定します。

継承元 **AnnotationBase**
型 **boolean**

name

注釈の名前を取得または設定します。

継承元 **AnnotationBase**
型 **string**

offset

point からの注釈のオフセットを取得または設定します。

継承元 **AnnotationBase**
型 **Point**

point

注釈のポイントを取得または設定します。 ポイントの座標は、**attachment** プロパティに依存します。 詳細は、**AnnotationAttachment** を参照してください。

継承元 **AnnotationBase**
型 **DataPoint**

pointIndex

注釈のデータポイントインデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

position

注釈の位置を取得または設定します。 位置は、**point** に対する相対的な位置です。

継承元 **AnnotationBase**
型 **AnnotationPosition**

seriesIndex

注釈のデータ系列インデックスを取得または設定します。 **attachment**プロパティがDataIndexに設定されている場合にのみ適用されます。

継承元 **AnnotationBase**
型 **number**

style

注釈のスタイルを取得または設定します。

継承元 **AnnotationBase**
型 **any**

text

注釈のテキストを取得または設定します。

継承元 **Text**
型 **string**

tooltip

注釈のツールチップを取得または設定します。

継承元 **AnnotationBase**
型 **string**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'items'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

destroy

```
destroy(): void
```

この注釈を破棄します。

継承元 **AnnotationBase**
戻り値 **void**

render

```
render(engine: IRenderEngine): void
```

この注釈をレンダリングします。

パラメーター

- **engine**: **IRenderEngine**
  注釈をレンダリングするためのエンジン。

継承元 **AnnotationBase**
戻り値 **void**

wijmo/wijmo.angular2.chart.finance モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.finance**

wijmo.chart.financeモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.chart.financeは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjFinance from 'wijmo/wijmo.angular2.chart.finance';

@Component({
    directives: [wjFinance.WjFinancialChart, wjFinance.WjFinancialChartSeries],
    template: `
      <wj-financial-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-financial-chart-series [binding]="'y'"></wj-financial-chart-series>
      </wj-financial-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjFinancialChart
- WjFinancialChartSeries

WjFinancialChart クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.finance
基本クラス FinancialChart
表示 ☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールに対応するAngular 2コンポーネント。

wj-financial-chartコンポーネントを使用して、Angular 2アプリケーションに**FinancialChart**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFinancialChartコンポーネントは、**FinancialChart**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-financial-chartコンポーネントには、 **WjFlexChartTrendLine** 、 **WjFlexChartMovingAverage** 、 **WjFlexChartYFunctionSeries** 、 **WjFlexChartParametricFunctionSeries** 、 **WjFlexChartWaterfall** 、 **WjFlexChartBoxWhisker** 、 **WjFlexChartAnimation** 、 **WjFlexChartAnnotationLayer** 、 **WjFlexChartFibonacci** 、 **WjFlexChartFibonacciArcs** 、 **WjFlexChartFibonacciFans** 、 **WjFlexChartFibonacciTimeZones** 、 **WjFlexChartAtr** 、 **WjFlexChartCci** 、 **WjFlexChartRsi** 、 **WjFlexChartWilliamsR** 、 **WjFlexChartMacd** 、 **WjFlexChartMacdHistogram** 、 **WjFlexChartStochastic** 、 **WjFlexChartBollingerBands** 、 **WjFlexChartEnvelopes** 、 **WjFinancialChartSeries** 、 **WjFlexChartRangeSelector** 、 **WjFlexChartGestures** 、 **WjFlexChartAxis** 、 **WjFlexChartLegend** 、 **WjFlexChartLineMarker** および **WjFlexChartPlotArea**、 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- interpolateNulls
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- lostFocusNg
- options

- palette
- plotAreas
- plotMargin
- renderedNg
- renderingNg
- rightToLeft
- selection
- selectionChangedNg
- selectionMode
- series
- seriesVisibilityChangedNg
- symbolSize
- tooltip
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FinancialChart
```

FlexChart クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  このコントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **FinancialChart**
戻り値 **FinancialChart**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

axes

Axis オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

axisX

メインのX軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

- axisY

メインのY軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

- binding

Yの値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

- bindingX

Xデータ値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

- chartType

作成する株価チャートのタイプを取得または設定します。

継承元 **FinancialChart**
型 **FinancialChartType**

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

- dataLabel

ポイントのデータラベルを取得または設定します。

継承元 **FlexChartCore**
型 **DataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **FlexChartCore**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine** 、描画する要素を記述する**HitTestInfo** パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

- itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

- legendToggle

凡例項目をクリックしたときにチャート上の系列の表示/非表示を切り替えるかどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 **FlexChartCore**
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

options

さまざまなチャートオプションを取得または設定します。

以下のオプションがサポートされます。

kagi.fields：カギ足チャートで使用される**DataFields** を指定します。デフォルト値はDataFields.Closeです。

kagi.rangeMode：カギ足チャートで使用される**RangeMode** を指定します。デフォルト値はRangeMode.Fixedです。

kagi.reversalAmount：カギ足チャートの反転量を指定します。デフォルト値は14です。

```
chart.options = {
  kagi: {
     fields: wijmo.chart.finance.DataFields.Close,
     rangeMode: wijmo.chart.finance.RangeMode.Fixed,
     reversalAmount: 14
  }
}
```

lineBreak.newLineBreaks:：ラインブレイクチャートで、何個のボックスを 比較してから新しいボックスを描画するかを 取得または設定します。デフォルト値は3です。

```
chart.options = {
  lineBreak: { newLineBreaks: 3 }
}
```

renko.fields：練行足チャートで使用される**DataFields** を指定します。デフォルト値はDataFields.Closeです。

renko.rangeMode：練行足チャートで使用される**RangeMode** を指定します。デフォルト値はRangeMode.Fixedです。

renko.boxSize：練行足チャートのボックスサイズを指定します。デフォルト値は14です。

```
chart.options = {
  renko: {
     fields: wijmo.chart.finance.DataFields.Close,
     rangeMode: wijmo.chart.finance.RangeMode.Fixed,
     boxSize: 14
  }
}
```

継承元　　**FinancialChart**
型　　**any**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string**[]

plotAreas

PlotArea オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **PlotAreaCollection**

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selection

選択されているチャート系列を取得または設定します。

継承元 **FlexChartCore**
型 **SeriesBase**

selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

series

Series オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

seriesVisibilityChangedNg

プログラムによるアクセスに使用されるWijmo **seriesVisibilityChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **seriesVisibilityChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

symbolSize

このFlexChart のすべてのSeriesオブジェクトに使用されるシンボルのサイズを取得または設定します。

このプロパティは、各Series オブジェクトのsymbolSizeプロパティによってオーバーライドできます。

The default value for this property is **10** pixels.

継承元 FlexChartCore
型 number

tooltip

チャートのTooltip オブジェクトを取得します。

ツールチップの内容は、次のパラメータを含むテンプレートを使用して生成されます。

- **propertyName**： このポイントによって表されるデータオブジェクトの任意のプロパティ。
- **seriesName**： データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**： データポイントのインデックス。
- **value**： データポイントの**値**（**FlexChart** の場合はy値、**FlexPie** の場合は項目値）。
- **x**： データポイントの**x**値（FlexChartのみ）。
- **y**： データポイントの**y**値（FlexChartのみ）。
- **name**： データポイントの**名前**（**FlexChart** の場合はx値、**FlexPie** の場合は凡例エントリ）。

テンプレートを変更するには、ツールチップのコンテンツプロパティに新しい値を割り当てます。 次に例を示します。

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

チャートのツールチップを無効にできます。それには、テンプレートを空の文字列に設定します。

tooltip プロパティを使用して、次のように、**showDelay** や**hideDelay** などの ツールチップパラメータをカスタマイズすることもできます。

```
chart.tooltip.showDelay = 1000;
```

詳細とオプションについては、**ChartTooltip** プロパティを参照してください。

継承元 FlexChartCore
型 ChartTooltip

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 string

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Point をデータ座標からコントロール座標に変換します。

パラメーター

- **pt**: **any**
  データ座標の**Point** 、またはデータ座標のポイントのX座標。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

seriesVisibilityChanged イベントを発生させます。

パラメーター

- **e**: **SeriesEventArgs**
  イベントデータを含む**SeriesEventArgs** オブジェクト。

継承元 **FlexChartCore**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

pointToData

```
pointToData(pt: any, y?: number): Point
```

Point をコントロール座標からチャートデータ座標に変換します。

パラメーター

- **pt**: **any**
  変換するポイント（コントロール座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**`ImageFormat`** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **`FlexChartBase`**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **`FlexChartBase`**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

seriesVisibilityChanged

系列の表示/非表示が変更されたときに発生します。たとえば、legendToggleプロパティをtrueに設定したり、ユーザーが凡例をクリックしたときです。

継承元 **FlexChartCore**
引数 **SeriesEventArgs**

WjFinancialChartSeries クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.finance
基本クラス FinancialSeries
表示 ☐継承されたメンバー ☐イベント発生元

FinancialSeries コントロールに対応するAngular 2コンポーネント。

wj-financial-chart-seriesコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-financial-chart-seriesコンポーネントを使用して、Angular 2アプリケーションに**FinancialSeries**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFinancialChartSeriesコンポーネントは、**FinancialSeries**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-financial-chart-series**コンポーネントには、 **WjFlexChartAxis** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect

- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): SeriesBase
```

SeriesBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **SeriesBase**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- chartType

特定の系列のチャートタイプを取得または設定します。これはチャート全体に設定されたチャートタイプをオーバーライドします。ColumnVolume、EquiVolume、CandleVolume、ArmsCandleVolumeの各チャートタイプはサポートされていない点に注意してください。これらのチャートタイプは**FinancialChart** に対して設定する必要があります

継承元 **FinancialSeries**
型 **FinancialChartType**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **FinancialSeries**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

wijmo/wijmo.angular2.chart.finance.analytics モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.chart.finance.analytics**

wijmo.chart.finance.analyticsモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.chart.finance.analyticsは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjFinanceAnalitics from 'wijmo/wijmo.angular2.chart.finance.analytics';
```

クラス

- WjFlexChartAtr
- WjFlexChartBollingerBands
- WjFlexChartCci
- WjFlexChartEnvelopes
- WjFlexChartFibonacci
- WjFlexChartFibonacciArcs
- WjFlexChartFibonacciFans
- WjFlexChartFibonacciTimeZones
- WjFlexChartMacd
- WjFlexChartMacdHistogram
- WjFlexChartRsi
- WjFlexChartStochastic
- WjFlexChartWilliamsR

WjFlexChartAtr クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.finance.analytics**
基本クラス **ATR**
表示 ☐継承されたメンバー ☐イベント発生元

ATR コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-atrコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-atrコンポーネントを使用して、Angular 2アプリケーションに**ATR**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjFlexChartAtrコンポーネントは、**ATR**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement

- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): ATR
```

ATR クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **ATR**
戻り値 **ATR**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

計算の期間を整数値として取得または設定します。

継承元 **SingleOverlayIndicatorBase**
型 **any**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartBollingerBands クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.finance.analytics**
基本クラス **BollingerBands**
表示 ☐継承されたメンバー ☐イベント発生元

BollingerBands コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-bollinger-bandsコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-bollinger-bandsコンポーネントを使用して、Angular 2アプリケーションに**BollingerBands**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartBollingerBandsコンポーネントは、**BollingerBands**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- multiplier
- name
- period
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem

- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): BollingerBands
```

BollingerBands クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **BollingerBands**
戻り値 **BollingerBands**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- multiplier

標準偏差乗数を取得または設定します。

継承元 **BollingerBands**
型 **number**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

計算の期間を整数値として取得または設定します。

継承元 **BollingerBands**
型 **any**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartCci クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.finance.analytics**
基本クラス **CCI**
表示 ☐継承されたメンバー ☐イベント発生元

CCI コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-cciコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-cciコンポーネントを使用して、Angular 2アプリケーションに**CCI** コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjFlexChartCciコンポーネントは、**CCI** コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- constant
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect

- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): CCI
```

CCI クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **CCI**
戻り値 **CCI**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- constant

CCI計算の定数値を取得または設定します。デフォルト値は0.015です。

継承元 **CCI**
型 **number**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

計算の期間を整数値として取得または設定します。

継承元 **SingleOverlayIndicatorBase**
型 **any**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartEnvelopes クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.finance.analytics
基本クラス Envelopes
表示 ☐継承されたメンバー ☐イベント発生元

Envelopes コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-envelopesコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-envelopesコンポーネントを使用して、Angular 2アプリケーションに**Envelopes**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartEnvelopesコンポーネントは、**Envelopes**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- size
- style
- symbolMarker
- symbolSize
- symbolStyle
- type
- visibility
- wjProperty

メソッド

- created
- dataToPoint

- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Envelopes
```

Envelopes クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Envelopes**
戻り値 **Envelopes**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

計算の期間を整数値として取得または設定します。

継承元 **Envelopes**
型 **any**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

size

移動平均エンベロープのサイズを取得または設定します。デフォルト値は2.5%（0.025）です。

継承元 **Envelopes**
型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

type

エンベロープの移動平均の種類を取得または設定します。デフォルト値はSimpleです。

継承元 **Envelopes**
型 **MovingAverageType**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

WjFlexChartFibonacci クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.finance.analytics**
基本クラス **Fibonacci**
表示 ☐継承されたメンバー ☐イベント発生元

Fibonacci コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-fibonacciコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-fibonacciコンポーネントを使用して、Angular 2アプリケーションに**Fibonacci**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartFibonacciコンポーネントは、**Fibonacci**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- high
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- labelPosition
- legendElement
- levels
- low
- maxX
- minX
- name
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- uptrend
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Fibonacci
```

Fibonacci クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Fibonacci**
戻り値 **Fibonacci**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- high

Fibonacci ツールの高値を取得または設定します。

これを指定しない場合、高値は**itemsSource** によって提供されたデータ値に基づいて計算されます。

継承元 **Fibonacci**
型 **number**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

labelPosition

Fibonacci ツールのレベルのラベル位置を取得または設定します。

継承元 **Fibonacci**
型 **LabelPosition**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- levels

プロットに使用する水準の配列を取得または設定します。

デフォルト値は[0, 23.6, 38.2, 50, 61.8, 100]です。

継承元 **Fibonacci**
型 **number[]**

- low

Fibonacci ツールの安値を取得または設定します。

これを指定しない場合、安値は**itemsSource**によって提供されたデータ値に基づいて計算されます。

継承元 **Fibonacci**
型 **number**

- maxX

Fibonacci ツールのxの最大値を取得または設定します。

これを指定しない場合、x軸の現在の最大値が使用されます。 値は数値またはDateオブジェクトで指定できます。

継承元 **Fibonacci**
型 **any**

- minX

Fibonacci ツールのxの最小値を取得または設定します。

これを指定しない場合、x軸の現在の最小値が使用されます。 値は数値またはDateオブジェクトで指定できます。

継承元 **Fibonacci**
型 **any**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

uptrend

上昇トレンドの**Fibonacci** ツールを作成するかどうかを示す値を取得または設定します。

デフォルト値はtrue（上昇トレンド）です。この値がfalseの場合は、下降トレンドのレベルがプロットされます。

継承元 **Fibonacci**
型 **boolean**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 `SeriesBase`
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 `SeriesBase`
戻り値 `Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 `SeriesBase`
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

WjFlexChartFibonacciArcs クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.finance.analytics
基本クラス FibonacciArcs
表示 ☐継承されたメンバー ☐イベント発生元

FibonacciArcs コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-fibonacci-arcsコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-fibonacci-arcsコンポーネントを使用して、Angular 2アプリケーションに**FibonacciArcs**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartFibonacciArcsコンポーネントは、**FibonacciArcs**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- end
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- labelPosition
- legendElement
- levels
- name
- renderedNg
- renderingNg
- start
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): FibonacciArcs
```

FibonacciArcs クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  初期化データを含むJavaScriptオブジェクト。

継承元 **FibonacciArcs**
戻り値 **FibonacciArcs**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- end

ベースラインの終点の**DataPoint** を取得または設定します。

DataPoint のxの値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

他の一部のフィボナッチツールとは異なり、終点の**DataPoint** を定義しなかった場合、これは自動的に計算**not**。

継承元 **FibonacciArcs**
型 **DataPoint**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- labelPosition

FibonacciArcs ツールの水準の**LabelPosition** を取得または設定します。

継承元 **FibonacciArcs**
型 **LabelPosition**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

levels

プロットに使用する水準の配列を取得または設定します。

デフォルト値は[38.2, 50, 61.8]です。

継承元 **FibonacciArcs**
型 **number[]**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

start

ベースラインの始点の **DataPoint** を取得または設定します。

DataPoint のxの値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

他の一部のフィボナッチツールとは異なり、始点の**DataPointを定義しなかった場合**、これは自動的に計算**not**。

継承元 **FibonacciArcs**
型 **DataPoint**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartFibonacciFans クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.finance.analytics
基本クラス FibonacciFans
表示 ☐継承されたメンバー ☐イベント発生元

FibonacciFans コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-fibonacci-fansコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-fibonacci-fansコンポーネントを使用して、Angular 2アプリケーションに**FibonacciFans**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartFibonacciFansコンポーネントは、**FibonacciFans**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- end
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- labelPosition
- legendElement
- levels
- name
- renderedNg
- renderingNg
- start
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): FibonacciFans
```

FibonacciFans クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  初期化データを含むJavaScriptオブジェクト。

継承元 **FibonacciFans**
戻り値 **FibonacciFans**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- end

ベースラインの終点の**DataPoint** を取得または設定します。

設定されていない場合、始点の**DataPoint** は自動的に計算されます。 **DataPoint** のxの値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

継承元 **FibonacciFans**
型 **DataPoint**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- labelPosition

FibonacciFans ツールの水準の**LabelPosition** を取得または設定します。

継承元 **FibonacciFans**
型 **LabelPosition**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

levels

プロットに使用する水準の配列を取得または設定します。

デフォルト値は[0, 23.6, 38.2, 50, 61.8, 100]です。

継承元 **FibonacciFans**
型 **number[]**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

start

ベースラインの始点の **DataPoint** を取得または設定します。

設定されていない場合、始点の**DataPoint** は自動的に計算されます。 **DataPoint** のxの値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

継承元 **FibonacciFans**
型 **DataPoint**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartFibonacciTimeZones クラス

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.chart.finance.analytics**
基本クラス **FibonacciTimeZones**
表示 ☐継承されたメンバー ☐イベント発生元

FibonacciTimeZones コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-fibonacci-time-zonesコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-fibonacci-time-zonesコンポーネントを使用して、Angular 2アプリケーションに**FibonacciTimeZones**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartFibonacciTimeZonesコンポーネントは、**FibonacciTimeZones**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- endX
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- labelPosition
- legendElement
- levels
- name
- renderedNg
- renderingNg
- startX
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): FibonacciTimeZones
```

FibonacciTimeZones クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  初期化データを含むJavaScriptオブジェクト。

継承元 **FibonacciTimeZones**
戻り値 **FibonacciTimeZones**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

endX

タイムゾーンの終了位置のXデータポイントを取得または設定します。

設定されていない場合、終了位置のXデータポイントは自動的に計算されます。値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

継承元 **FibonacciTimeZones**
型 **any**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

labelPosition

FibonacciTimeZones ツールの水準の**LabelPosition** を取得または設定します。

継承元 **FibonacciTimeZones**
型 **LabelPosition**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

levels

プロットに使用する水準の配列を取得または設定します。

デフォルト値は[0, 1, 2, 3, 5, 8, 13, 21, 34]です。

継承元 **FibonacciTimeZones**
型 **number[]**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

startX

タイムゾーンの開始位置のXデータポイントを取得または設定します。

設定されていない場合、開始位置のXデータポイントは自動的に計算されます。値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

継承元 **FibonacciTimeZones**
型 **any**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartMacd クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.finance.analytics
基本クラス Macd
表示 ☐継承されたメンバー ☐イベント発生元

Macd コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-macdコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-macdコンポーネントを使用して、Angular 2アプリケーションに**Macd**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartMacdコンポーネントは、**Macd**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- fastPeriod
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- renderedNg
- renderingNg
- slowPeriod
- smoothingPeriod
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint

- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Macd
```

Macd クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Macd**
戻り値 **Macd**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- fastPeriod

MACDラインの高速の指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

slowPeriod

MACDラインの低速の指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

smoothingPeriod

シグナルラインの指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

styles

MACDラインとシグナルラインのスタイルを取得または設定します。

以下のオプションがサポートされています。

```
series.styles = {
  macdLine: {
     stroke: 'red',
     strokeWidth: 1
  },
  signalLine: {
     stroke: 'green',
     strokeWidth: 1
  },
}
```

継承元 **Macd**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartMacdHistogram クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.finance.analytics**
基本クラス **MacdHistogram**
表示 ☐継承されたメンバー ☐イベント発生元

MacdHistogram コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-macd-histogramコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-macd-histogramコンポーネントを使用して、Angular 2アプリケーションに**MacdHistogram**コントロールトを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartMacdHistogramコンポーネントは、**MacdHistogram**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- fastPeriod
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- renderedNg
- renderingNg
- slowPeriod
- smoothingPeriod
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint

- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): MacdHistogram
```

MacdHistogram クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **MacdHistogram**
戻り値 **MacdHistogram**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- fastPeriod

MACDラインの高速の指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

slowPeriod

MACDラインの低速の指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

smoothingPeriod

シグナルラインの指数移動平均の期間を取得または設定します。

継承元 **MacdBase**
型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。
- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元	**SeriesBase**
引数	**EventArgs**

WjFlexChartRsi クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.finance.analytics**
基本クラス **RSI**
表示 ☐継承されたメンバー ☐イベント発生元

RSI コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-rsiコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-rsiコンポーネントを使用して、Angular 2アプリケーションに**RSI**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjFlexChartRsiコンポーネントは、**RSI**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement

- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): RSI
```

RSI クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **RSI**
戻り値 **RSI**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

計算の期間を整数値として取得または設定します。

継承元 **SingleOverlayIndicatorBase**
型 **any**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartStochastic クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.finance.analytics
基本クラス Stochastic
表示 ☐継承されたメンバー ☐イベント発生元

Stochastic コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-stochasticコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-stochasticコンポーネントを使用して、Angular 2アプリケーションに**Stochastic**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartStochasticコンポーネントは、**Stochastic**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- dPeriod
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- kPeriod
- legendElement
- name
- renderedNg
- renderingNg
- smoothingPeriod
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): Stochastic
```

Stochastic クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **Stochastic**
戻り値 **Stochastic**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- dPeriod

%D単純移動平均の期間を取得または設定します。

継承元 **Stochastic**
型 **number**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- kPeriod

%K計算の期間を取得または設定します。

継承元 **Stochastic**
型 **number**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

smoothingPeriod

フル%Kの平滑化期間を取得または設定します。

継承元 **Stochastic**
型 **number**

style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

styles

%Kラインと%Dラインのスタイルを取得または設定します。

以下のオプションがサポートされています。

```
series.styles = {
  kLine: {
     stroke: 'red',
     strokeWidth: 1
  },
  dLine: {
     stroke: 'green',
     strokeWidth: 1
  },
}
```

継承元 **Stochastic**
型 **any**

symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **rect**: **Rect**
  凡例項目の位置。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

- **index**: **number**
  レンダリングする系列のインデックス。

- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

WjFlexChartWilliamsR クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.finance.analytics**
基本クラス **WilliamsR**
表示 ☐継承されたメンバー ☐イベント発生元

WilliamsR コントロールに対応するAngular 2コンポーネント。

wj-flex-chart-williams-rコンポーネントは、 **WjFinancialChart** コンポーネントに含める必要があります。

wj-flex-chart-williams-rコンポーネントを使用して、Angular 2アプリケーションに**WilliamsR**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexChartWilliamsRコンポーネントは、**WilliamsR**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem
- getDataRect

- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): WilliamsR
```

WilliamsR クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **WilliamsR**
戻り値 **WilliamsR**

プロパティ

- altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

- axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

- hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

- legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

- name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- period

計算の期間を整数値として取得または設定します。

継承元 **SingleOverlayIndicatorBase**
型 **any**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

wijmo/wijmo.angular2.chart.hierarchical モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.chart.hierarchical**

wijmo.chart.hierarchicalモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.chart.hierarchicalは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjHierarchical from 'wijmo/wijmo.angular2.chart.hierarchical';

@Component({
    directives: [wjHierarchical.WjSunburst],
    template: `
      <wj-sunburst [itemsSource]="data" [binding]="'y'" [bindingX]="'x'">
      </wj-sunburst>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjSunburst
- WjTreeMap

WjSunburst クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.hierarchical**
基本クラス **Sunburst**
表示 ☐継承されたメンバー ☐イベント発生元

Sunburst コントロールに対応するAngular 2コンポーネント。

wj-sunburstコンポーネントを使用して、Angular 2アプリケーションに**Sunburst**コントロールを追加します。 Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjSunburstコンポーネントは、**Sunburst**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-sunburstコンポーネントには、**WjFlexChartLegend** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- binding
- bindingName
- childItemsPath
- collectionView
- dataLabel
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- innerRadius
- isAnimated
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- lostFocusNg
- offset
- palette
- plotMargin
- renderedNg
- renderingNg
- reversed
- rightToLeft
- selectedIndex
- selectedItemOffset
- selectedItemPosition

- selectionChangedNg
- selectionMode
- startAngle
- tooltip
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FlexPie
```

FlexPie クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  このコントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavascriptオブジェクト。

継承元 **FlexPie**
戻り値 **FlexPie**

プロパティ

binding

チャート値を含むプロパティの名前を取得または設定します。

継承元 **FlexPie**
型 **string**

bindingName

データ項目の名前を含むプロパティの名前を取得または設定します。この名前は配列または文字列である必要があります。

継承元 **Sunburst**
型 **any**

childItemsPath

階層化データの子項目の生成に使用される1つ以上のプロパティの名前を 取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `'items'` ）。

項目が異なる名前を持つ異なるレベルの子項目の場合は、 このプロパティに、各レベルの子項目を含むプロパティの名前から成る 配列を設定します （たとえば、 `[ 'accounts', 'checks', 'earnings' ]` ）。

継承元 **Sunburst**
型 **any**

collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

dataLabel

ポイントのデータラベルを取得または設定します。

継承元 **FlexPie**
型 **PieDataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

innerRadius

パイの内側半径のサイズを取得または設定します。

内側半径はパイ半径に対する割合として測定されます。

このプロパティのデフォルト値はゼロです（つまり、円グラフになります）。このプロパティをゼロより大きい値に設定すると、円グラフの中央に穴が開きます（これをドーナツグラフと呼びます）。

The default value for this property is **0**

継承元 **FlexPie**
型 **number**

isAnimated

項目が選択されたときにアニメーションを使用するかどうかを示す値を取得または設定します。

selectedItemPosition プロパティおよび**selectionMode** プロパティも参照してください。

The default value for this property is **false**.

継承元 **FlexPie**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

offset

スライスのパイ中心からのオフセットを取得または設定します。

オフセットはパイ半径に対する割合として測定されます。

The default value for this property is **0**.

継承元 **FlexPie**
型 **number**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string**[]

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

reversed

角度を反転（反時計回り）するかどうかを決定する値を取得または設定します。

デフォルト値はfalseです。この場合、角度は時計回りに測定されます。

The default value for this property is **false**.

継承元 **FlexPie**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectedIndex

選択されたスライスのインデックスを取得または設定します。

継承元 **FlexPie**
型 **number**

selectedItemOffset

選択されたスライスのパイ中心からのオフセットを取得または設定します。

オフセットはパイ半径に対する割合として測定されます。

The default value for this property is **0**.

継承元 **FlexPie**
型 **number**

selectedItemPosition

選択されたスライスの位置を取得または設定します。

このプロパティを'None'以外の値に設定すると、スライスを選択したときに円グラフが回転します。

円グラフをクリックしたときにスライスが選択されるようにするには、 **selectionMode** プロパティを'Point'に設定する必要があります。

The default value for this property is **Position.None**.

継承元 **FlexPie**
型 **Position**

selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

startAngle

パイスライスの開始角度（度単位）を取得または設定します。

角度は9時の位置から時計回りに測定されます。

The default value for this property is **0**.

継承元 **FlexPie**
型 **number**

tooltip

チャートの**Tooltip** を取得します。

継承元 **FlexPie**
型 **ChartTooltip**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は''です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexPie**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**ImageFormat** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **FlexChartBase**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **FlexChartBase**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

WjTreeMap クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.hierarchical**
基本クラス **TreeMap**
表示 ☐継承されたメンバー ☐イベント発生元

TreeMap コントロールに対応するAngular 2コンポーネント。

wj-tree-mapコンポーネントを使用して、Angular 2アプリケーションに**TreeMap**コントロールを追加できます。 Angular 2マークアップの構文については、「Angular 2マークアップ」を参照してください。 **WjTreeMap**コンポーネントは、**TreeMap**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- binding
- bindingName
- childItemsPath
- collectionView
- dataLabel
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- lostFocusNg
- maxDepth
- palette
- plotMargin
- renderedNg
- renderingNg
- rightToLeft
- selectionChangedNg
- selectionMode
- tooltip
- type
- wjModelProperty

メソッド

- addEventListener

- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged

コンストラクタ

constructor

```
constructor(element: any, options?): TreeMap
```

TreeMap クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavascriptオブジェクト。

継承元 **TreeMap**
戻り値 **TreeMap**

プロパティ

binding

チャートの値を含むデータ項目のプロパティの名前を取得または設定します。

連結プロパティは、他のノードの値と比較してノードのサイズを計算するために使用されます。 このプロパティには、数値データを設定する必要があります。

継承元 **TreeMap**
型 **string**

bindingName

データ項目の名前を含むプロパティの名前を取得または設定します。 bindingNameプロパティは、ノードの名前を表示するために使用されます。この名前は配列または文字列である必要があります。

継承元 **TreeMap**
型 **any**

childItemsPath

階層化データの子項目の生成に使用される1つ以上のプロパティの名前を 取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `'items'` ）。

項目が異なる名前を持つ異なるレベルの子項目の場合は、 このプロパティに、各レベルの子項目を含むプロパティの名前から成る 配列を設定します（たとえば、 `[ 'accounts', 'checks', 'earnings' ]` ）。

継承元 **TreeMap**
型 **any**

collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

- dataLabel

ツリーマップの**DataLabel** を取得または設定します。

継承元 **TreeMap**
型 **DataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の**gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine**、描画する要素を記述する**HitTestInfo**パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView**オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

maxDepth

ドロップダウンに表示する項目の最大数を取得または設定します。 これらのレベルが現在の平面に平坦化されます。 ツリーマップのレベルがこの値よりも高い場合、ユーザーは上下に移動する必要があります。

継承元 **TreeMap**
型 **number**

palette

ツリーマップで使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

または、titleColor、maxColor、minColorを別々に含みます。次に例を示します。

```
chart.palette = [{
     titleColor: '#00277d',
     maxColor: 'rgba(0,39,125,0.7)',
     minColor: 'rgba(168,187,230,0.7)'
 }, {
     titleColor: '#7d1f00',
     maxColor: 'rgba(125,21,0,0.7)',
     minColor: 'rgba(230,183,168,0.7)'
 }, {
     titleColor: '#007d27',
     maxColor: 'rgba(0,125,39,0.7)',
     minColor: 'rgba(168,230,188,0.7)'
 }];
```

継承元 **TreeMap**
型 **string**[]

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectionMode

selectionModeは、TreeMapコントロールでは機能しません。

継承元 **TreeMap**
型 **SelectionMode**

- tooltip

チャートの**Tooltip** を取得します。

継承元 **TreeMap**
型 **ChartTooltip**

- type

ツリーマップの**TreeMapType** を取得または設定します。

継承元 **TreeMap**
型 **TreeMapType**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は''です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **TreeMap**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**ImageFormat** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **FlexChartBase**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **FlexChartBase**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

wijmo/wijmo.angular2.chart.radar モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.radar**

wijmo.chart.radarモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.chart.radarは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjRadar from 'wijmo/wijmo.angular2.chart.radar';

@Component({
    directives: [wjRadar.WjFlexRadar, wjRadar.WjFlexRadarSeries],
    template: `
      <wj-flex-radar [itemsSource]="data" [bindingX]="'x'">
          <wj-flex-radar-series [binding]="'y'"></wj-flex-radar-series>
      </wj-flex-radar>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjFlexRadar
- WjFlexRadarAxis
- WjFlexRadarSeries

WjFlexRadar クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.radar**
基本クラス **FlexRadar**
表示 ☐継承されたメンバー ☐イベント発生元

FlexRadar コントロールに対応するAngular 2コンポーネント。

wj-flex-radarコンポーネントを使用して、Angular 2アプリケーションに**FlexRadar**コントロールを追加します。 Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjFlexRadarコンポーネントは、**FlexRadar**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-flex-radarコンポーネントには、**WjFlexChartAnimation** 、 **WjFlexRadarAxis** 、 **WjFlexRadarSeries** 、および**WjFlexChartLegend** を子コンポーネントとして含めることができます。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- interpolateNulls
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- lostFocusNg
- palette
- plotAreas
- plotMargin
- renderedNg
- renderingNg
- reversed

- reversed
- rightToLeft
- selection
- selectionChangedNg
- selectionMode
- series
- seriesVisibilityChangedNg
- stacking
- startAngle
- symbolSize
- tooltip
- totalAngle
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

コンストラクタ

constructor

```
constructor(element: any, options?): FlexRadar
```

FlexRadar クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  このコントロールをホストするDOM要素またはホスト要素のセレクタ（'#theCtrl'など）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **FlexRadar**
戻り値 **FlexRadar**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

axes

Axis オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

axisX

メインのX軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

- axisY

メインのY軸を取得または設定します。

継承元 **FlexChartCore**
型 **Axis**

- binding

Yの値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

- bindingX

Xデータ値を含むプロパティの名前を取得または設定します。

継承元 **FlexChartCore**
型 **string**

- chartType

作成するレーダーチャートのタイプを取得または設定します。

継承元 **FlexRadar**
型 **RadarChartType**

- collectionView

チャートデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **FlexChartBase**
型 **ICollectionView**

- dataLabel

ポイントのデータラベルを取得または設定します。

継承元 **FlexChartCore**
型 **DataLabel**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 **FlexChartBase**
型 **string**

headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 **FlexChartBase**
型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **FlexChartCore**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

チャート要素の外観をカスタマイズするための項目書式設定関数を取得または設定します。

指定されている場合、関数は、チャート上に要素を描画する**IRenderEngine** 、描画する要素を記述する**HitTestInfo** パラメータ、および項目のデフォルトの描画を提供する関数の3つのパラメータを受け取る必要があります。

次に例を示しています。

```
itemFormatter: function (engine, hitTestInfo, defaultRenderer) {
  var ht = hitTestInfo,
      binding = 'downloads';

  // 正しい系列/要素であることを確認します
  if (ht.series.binding == binding && ht.pointIndex > 0 &&
      ht.chartElement == wijmo.chart.ChartElement.SeriesSymbol) {

    // 現在値と前の値を取得します
    var chart = ht.series.chart,
        items = chart.collectionView.items,
        valNow = items[ht.pointIndex][binding],
        valPrev = items[ht.pointIndex - 1][binding];

    // 値が増加している場合は行を追加します
    if (valNow > valPrev) {
      var pt1 = chart.dataToPoint(ht.pointIndex, valNow),
          pt2 = chart.dataToPoint(ht.pointIndex - 1, valPrev);
      engine.drawLine(pt1.x, pt1.y, pt2.x, pt2.y, null, {
        stroke: 'gold',
        strokeWidth: 6
      });
    }
  }

  // 要素を通常通りに描画します。
  defaultRenderer();
}
```

サンプル

Show me (https://jsfiddle.net/Wijmo5/rptz23nL)

継承元 **FlexChartBase**
型 **Function**

itemsSource

チャートの作成に使用されるデータを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **FlexChartBase**
型 **any**

legend

チャートの凡例を取得または設定します。

継承元 **FlexChartBase**
型 **Legend**

legendToggle

凡例項目をクリックしたときにチャート上の系列の表示/非表示を切り替えるかどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 **FlexChartCore**
型 **boolean**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

palette

系列の表示に使用されるデフォルトの色の配列を取得または設定します。

この配列には、CSS色を表す文字列が含まれます。次に例を示します。

```
// 名前で指定された色を使用します
chart.palette = ['red', 'green', 'blue'];

// または、RGBA値で指定された色を使用します
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Palettes クラスにある事前定義されたパレットのセットを使用できます。次に例を示します。

```
chart.palette = wijmo.chart.Palettes.coral;
```

継承元 **FlexChartBase**
型 **string**[]

plotAreas

PlotArea オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **PlotAreaCollection**

plotMargin

プロットマージン（ピクセル単位）を取得または設定します。

プロットマージンは、コントロールの端からプロット領域の端までの領域を表します。

デフォルトでは、この値は軸ラベルに必要なスペースに基づいて自動的に計算されますが、コントロール内のプロット領域の位置を精密に制御したい場合（たとえば、複数のチャートコントロールをページ上に整列させるときなど）はこれをオーバーライドできます。

このプロパティは数値またはCSSスタイルのマージン指定に設定できます。例:

```
// すべての側のプロットマージンを20ピクセルに設定します。
chart.plotMargin = 20;

// 上側、右側、下側、左側のプロットマージンを設定します。
chart.plotMargin = '10 15 20 25';

// 上側/下側（10px）および左側/右側（20px）のプロットマージンを設定します。
chart.plotMargin = '10 20';
```

継承元 **FlexChartBase**
型 **any**

renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

reversed

角度を反転（反時計回り）するかどうかを決定する値を取得または設定します。

デフォルト値はfalseです。この場合、角度は時計回りに測定されます。

継承元 **FlexRadar**
型 **boolean**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selection

選択されているチャート系列を取得または設定します。

継承元 **FlexChartCore**
型 **SeriesBase**

- selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectionMode

ユーザーがチャートをクリックしたときに何が選択されるかを示す列挙値を取得または設定します。

The default value for this property is **SelectionMode.None**.

継承元 **FlexChartBase**
型 **SelectionMode**

- series

Series オブジェクトのコレクションを取得します。

継承元 **FlexChartCore**
型 **ObservableArray**

- seriesVisibilityChangedNg

プログラムによるアクセスに使用されるWijmo **seriesVisibilityChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **seriesVisibilityChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- stacking

系列オブジェクトを積み重ねるかどうかを決定する値と、積み重ねる場合はその方法を取得または設定します。

継承元 **FlexRadar**
型 **Stacking**

- startAngle

レーダーの開始角度（度単位）を取得または設定します。

角度は、12時の位置から時計回りに測定されます。

継承元 **FlexRadar**
型 **number**

symbolSize

このFlexChart のすべてのSeriesオブジェクトに使用されるシンボルのサイズを取得または設定します。

このプロパティは、各Series オブジェクトのsymbolSizeプロパティによってオーバーライドできます。

The default value for this property is **10** pixels.

継承元 FlexChartCore
型 number

tooltip

チャートのTooltip オブジェクトを取得します。

ツールチップの内容は、次のパラメータを含むテンプレートを使用して生成されます。

- **propertyName**： このポイントによって表されるデータオブジェクトの任意のプロパティ。
- **seriesName**： データポイントを含む系列の名前（FlexChartのみ）。
- **pointIndex**： データポイントのインデックス。
- **value**： データポイントの**値**（**FlexChart** の場合はy値、**FlexPie** の場合は項目値）。
- **x**： データポイントの**x**値（FlexChartのみ）。
- **y**： データポイントの**y**値（FlexChartのみ）。
- **name**： データポイントの**名前**（**FlexChart** の場合はx値、**FlexPie** の場合は凡例エントリ）。

テンプレートを変更するには、ツールチップのコンテンツプロパティに新しい値を割り当てます。 次に例を示します。

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

チャートのツールチップを無効にできます。それには、テンプレートを空の文字列に設定します。

tooltip プロパティを使用して、次のように、**showDelay** や**hideDelay** などの ツールチップパラメータをカスタマイズすることもできます。

```
chart.tooltip.showDelay = 1000;
```

詳細とオプションについては、**ChartTooltip** プロパティを参照してください。

継承元 FlexChartCore
型 ChartTooltip

totalAngle

レーダーの合計角度（度単位）を取得または設定します。デフォルト値は360です この値は、0より大きく、360以下である必要があります。

継承元 FlexRadar
型 number

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 string

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Point をデータ座標からコントロール座標に変換します。

パラメーター

- **pt**: **any**
  データ座標の**Point** 、またはデータ座標のポイントのX座標。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRendered

```
onRendered(e: RenderEventArgs): void
```

rendered イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onRendering

```
onRendering(e: RenderEventArgs): void
```

rendering イベントを発生させます。

パラメーター

- **e**: **RenderEventArgs**
  チャートのレンダリングに使用される**RenderEventArgs** オブジェクト。

継承元 **FlexChartBase**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexChartBase**
戻り値 **void**

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

seriesVisibilityChanged イベントを発生させます。

パラメーター

- **e**: **SeriesEventArgs**
  イベントデータを含む**SeriesEventArgs** オブジェクト。

継承元 **FlexChartCore**
戻り値 **void**

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

ページ座標をコントロール座標に変換します。

パラメーター

- **pt**: **any**
  ページ座標のポイントまたはページ座標のx値。

- **y**: **number** OPTIONAL
  ページ座標のy値。 ptが数値型の場合、値は数値である必要があります。 ただし、ptがPoint型の場合は、yパラメータはオプションになります。

継承元 **FlexChartBase**
戻り値 **Point**

pointToData

```
pointToData(pt: any, y?: number): Point
```

Point をコントロール座標からチャートデータ座標に変換します。

パラメーター

- **pt**: **any**
  変換するポイント（コントロール座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **FlexChartCore**
戻り値 **Point**

refresh

```
refresh(fullUpdate?: boolean): void
```

チャートを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示す値。

継承元 **FlexChartBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**`ImageFormat`** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **`FlexChartBase`**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **`FlexChartBase`**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **`Control`**
引数 **`EventArgs`**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

rendered

チャートのレンダリングが完了した後に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

rendering

チャートデータのレンダリングが開始される前に発生します。

継承元 **FlexChartBase**
引数 **RenderEventArgs**

selectionChanged

プログラムコードまたはユーザーがチャートをクリックしたことによって選択が変更された後に発生します。これは、たとえば詳細情報を示すテキストボックスを更新して現在の選択を表示するような場合に役立ちます。

継承元 **FlexChartBase**
引数 **EventArgs**

seriesVisibilityChanged

系列の表示/非表示が変更されたときに発生します。たとえば、legendToggleプロパティをtrueに設定したり、ユーザーが凡例をクリックしたときです。

継承元 **FlexChartCore**
引数 **SeriesEventArgs**

WjFlexRadarAxis クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.chart.radar**
基本クラス **FlexRadarAxis**
表示 ☐継承されたメンバー ☐イベント発生元

FlexRadarAxis コントロールに対応するAngular 2コンポーネント。

wj-flex-radar-axisコンポーネントは、 **WjFlexRadar** または、 **WjFlexRadarSeries** コンポーネントの子である必要があります。

wj-flex-radar-axisコンポーネントを使用して、Angular 2アプリケーションに**FlexRadarAxis**コントロールを追加します。 Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexRadarAxisコンポーネントは、**FlexRadarAxis**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- actualMax
- actualMin
- axisLine
- axisType
- binding
- format
- hostElement
- initialized
- isInitialized
- itemFormatter
- itemsSource
- labelAlign
- labelAngle
- labelPadding
- labels
- logBase
- majorGrid
- majorTickMarks
- majorUnit
- max
- min
- minorGrid
- minorTickMarks
- minorUnit
- name
- origin
- overlappingLabels
- plotArea
- position
- rangeChangedNg
- reversed
- title
- wjProperty

メソッド

- convert
- convertBack
- created
- onRangeChanged

イベント

- rangeChanged

コンストラクタ

constructor

```
constructor(position?: Position): Axis
```

Axis クラスの新しいインスタンスを初期化します。

パラメーター

- **position**: **Position** OPTIONAL
  チャート上での軸の位置。

継承元 **Axis**
戻り値 **Axis**

プロパティ

actualMax

実際の軸の最大を取得します。

これは、数値またはDateオブジェクト（時間ベースのデータの場合）を返します。

継承元 **Axis**
型 **any**

actualMin

実際の軸の最小を取得します。

これは、数値またはDateオブジェクト（時間ベースのデータの場合）を返します。

継承元 **Axis**
型 **any**

axisLine

軸線が表示されるかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Axis**
型 **boolean**

axisType

軸タイプを取得します。

継承元 **Axis**
型 **AxisType**

binding

軸ラベルで使用する**itemsSource** プロパティの カンマ区切りのプロパティ名を取得または設定します。

最初の名前は軸の値を指定し、2番目は対応する軸ラベルを表します。デフォルト値は'value,text'です。

継承元 **Axis**
型 **string**

format

軸ラベルに使用される書式文字列を取得または設定します （**Globalize** を参照）。

継承元 **Axis**
型 **string**

hostElement

軸のホスト要素を取得します。

継承元 **Axis**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemFormatter

軸ラベルのitemFormatter関数を取得または設定します。

指定された場合、関数は次の2つのパラメータをとります。

- **render engine**：ラベルの書式設定に使用される**IRenderEngine** オブジェクト。
- **current label**：以下のプロパティを含むオブジェクト。
  - **value**：書式設定する軸ラベルの値。
  - **text**：ラベルに使用するテキスト。
  - **pos**：ラベルがレンダリングされる コントロール座標内の位置。
  - **cls**：ラベルに適用されるCSSクラス。

この関数は、プロパティが変更されるラベルの ラベルパラメータを返します。

次に例を示します。

```
chart.axisY.itemFormatter = function(engine, label) {
    if (label.val > 5){
        engine.textFill = 'red'; // 赤色のテキスト
        label.cls = null; // デフォルトのCSSなし
     }
    return label;
}
```

継承元 **Axis**
型 **Function**

itemsSource

軸ラベルの項目ソースを取得または設定します。

プロパティの名前は、**binding** プロパティによって指定されます。

次に例を示します。

```
// Axis.bindingのデフォルト値は'value,text'です
 chart.axisX.itemsSource = [ { value:1, text:'one' }, { value:2, text:'two' } ];
```

<

継承元 **Axis**
型 **any**

labelAlign

ラベルの配置を取得または設定します。

デフォルトでは、ラベルは中央に配置されます。サポートされている値は、'left'および'right'（x軸の場合）と'top'および'bottom'（y軸の場合）です。

継承元 **Axis**
型 **string**

labelAngle

軸ラベルの回転角度を取得または設定します。

角度は度単位で測定されます。有効な値の範囲は-90～90です。

継承元 **Axis**
型 **number**

labelPadding

ラベルのパディングを取得または設定します。 The default value for this property is **5** pixels.

継承元 **Axis**
型 **number**

labels

軸ラベルを表示すかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Axis**
型 **boolean**

logBase

軸の対数の底を取得または設定します。

底が指定されていない場合、その軸は線形スケールになります。

logBase プロパティを使用すると、原点の周囲に集まっているデータが広がります。これは一部の金融データセットや経済データセットでよく見られます。

継承元 **Axis**
型 **number**

majorGrid

軸のグリッド線が表示されるかどうかを示す値を取得または設定します。

継承元 **Axis**
型 **boolean**

majorTickMarks

軸の目盛りマークの場所を取得または設定します。

継承元 **Axis**
型 **TickMark**

majorUnit

軸ラベル間の単位数を取得または設定します。

軸の値が日付の場合、単位は日数になります。

継承元 **Axis**
型 **number**

max

軸に表示される最大値を取得または設定します。

値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

The default value for this property is **null**, which causes the chart to calculate the maximum value based on the data.

継承元 **Axis**
型 **any**

min

軸に表示される最小値を取得または設定します。

値は数値またはDateオブジェクト（データが時間ベースの場合）で指定できます。

The default value for this property is **null**, which causes the chart to calculate the minimum value based on the data.

継承元 **Axis**
型 **any**

minorGrid

軸の副グリッド線が表示されるかどうかを示す値を取得または設定します。

継承元 **Axis**
型 **boolean**

minorTickMarks

小軸目盛りマークの場所を取得または設定します。

継承元 **Axis**
型 **TickMark**

minorUnit

軸の補助目盛間の単位数を取得または設定します。

軸の値が日付の場合、単位は日数になります。

継承元 **Axis**
型 **number**

name

軸の名前を取得または設定します。

継承元 **Axis**
型 **string**

origin

軸が直交軸と交差する位置の値を取得または設定します。

継承元 **Axis**
型 **number**

overlappingLabels

重なった軸ラベルの処理方法を示す値を取得または設定します。 The default value for this property is **OverlappingLabels.Auto**.

継承元 **Axis**
型 **OverlappingLabels**

plotArea

軸のプロットエリアを取得または設定します。

継承元 **Axis**
型 **PlotArea**

position

プロットエリアに対する軸の位置を取得または設定します。

継承元 **Axis**
型 **Position**

rangeChangedNg

プログラムによるアクセスに使用されるWijmo **rangeChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rangeChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

reversed

軸を反転させる（上から下方向、または右から左方向にする）かどうかを示す値を取得または設定します。 The default value for this property is **false**.

継承元 **Axis**
型 **boolean**

title

軸の横に表示されるタイトルのテキストを取得または設定します。

継承元 **Axis**
型 **string**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'axes'です。

型 **string**

メソッド

convert

```
convert(val: number, maxValue?: number, minValue?: number): number
```

指定した値をデータ座標からピクセル座標に変換します。

パラメーター

- **val**: **number**
  変換するデータ値。
- **maxValue**: **number** OPTIONAL
  データの最大値（オプション）。
- **minValue**: **number** OPTIONAL
  データの最小値（オプション）。

継承元 **Axis**
戻り値 **number**

convertBack

```
convertBack(val: number): number
```

指定した値をピクセル座標からデータ座標に変換します。

パラメーター

- **val**: **number**
  変換し戻すピクセル座標。

継承元 **Axis**
戻り値 **number**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

rangeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Axis**
戻り値 **void**

イベント

rangeChanged

軸範囲が変更されたときに発生します。

継承元 **Axis**
引数 **EventArgs**

WjFlexRadarSeries クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.chart.radar
基本クラス FlexRadarSeries
表示 ☐継承されたメンバー ☐イベント発生元

FlexRadarSeries コントロールに対応するAngular 2コンポーネント。

wj-flex-radar-seriesコンポーネントは、 **WjFlexRadar** コンポーネントの子である必要があります。

wj-flex-radar-seriesコンポーネントを使用して、Angular 2アプリケーションに**FlexRadarSeries**コントロールを追加します。 Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjFlexRadarSeriesコンポーネントは、**FlexRadarSeries**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-flex-radar-series コンポーネントには、 **WjFlexRadarAxis** を子コンポーネントとして含めることができます。

コンストラクタ

- constructor

プロパティ

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- initialized
- interpolateNulls
- isInitialized
- itemsSource
- legendElement
- name
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

メソッド

- created
- dataToPoint
- drawLegendItem

- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

イベント

- rendered
- rendering

コンストラクタ

constructor

```
constructor(options?: any): SeriesBase
```

SeriesBase クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **any** OPTIONAL
  オブジェクトの初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **SeriesBase**

プロパティ

altStyle

系列の代替スタイルを取得または設定します。このプロパティの値は、 横棒グラフ、縦棒グラフ、散布図の負の値に使用されたり、 ローソク足チャート、ラインブレイク、エクイボリュームなどの 株価チャートタイプの上昇値に使用されます。

このプロパティのデフォルト値はnullであり、系列でデフォルトのスタイルが使用されます。

継承元 **SeriesBase**
型 **any**

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

axisX

系列のx軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- axisY

系列のy軸を取得または設定します。

継承元 **SeriesBase**
型 **Axis**

- binding

系列のYの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- bindingX

系列のXの値を含むプロパティの名前を取得または設定します。

継承元 **SeriesBase**
型 **string**

- chart

この系列を所有する**FlexChart** オブジェクトを取得します。

継承元 **SeriesBase**
型 **FlexChartCore**

- chartType

特定の系列のチャートタイプを取得または設定します。これはチャート全体に設定されたチャートタイプをオーバーライドします。ColumnVolume、EquiVolume、CandleVolume、ArmsCandleVolumeの各チャートタイプはサポートされていない点に注意してください。これらのチャートタイプは**FinancialChart** に対して設定する必要があります

継承元 **FlexRadarSeries**
型 **RadarChartType**

- collectionView

この系列のデータを含む**ICollectionView** オブジェクトを取得します。

継承元 **SeriesBase**
型 **ICollectionView**

- cssClass

系列のCSSクラスを取得または設定します。

継承元 **SeriesBase**
型 **string**

hostElement

系列のホスト要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

interpolateNulls

データ内のnull値を補間するかどうかを決定する値を取得または設定します。

trueの場合は、隣接するポイントに基づいて、欠けているデータの値が補間されます。falseの場合は、null値のあるポイントで、線や領域に切れ目ができます。

The default value for this property is **false**.

継承元 **SeriesBase**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

itemsSource

系列データを含む配列または**ICollectionView** オブジェクトを取得または設定します。

継承元 **SeriesBase**
型 **any**

legendElement

系列の凡例要素を取得します。

継承元 **SeriesBase**
型 **SVGGElement**

name

系列の名前を取得または設定します。

系列名はチャートの凡例に表示されます。名前のない系列は凡例に表示されません。

継承元 **SeriesBase**
型 **string**

- renderedNg

プログラムによるアクセスに使用されるWijmo **rendered**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendered** Wijmoイベント名を使用してください。

型 **EventEmitter**

- renderingNg

プログラムによるアクセスに使用されるWijmo **rendering**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rendering** Wijmoイベント名を使用してください。

型 **EventEmitter**

- style

系列のスタイルを取得または設定します。

継承元 **SeriesBase**
型 **any**

- symbolMarker

系列の各データポイントに使用するマーカーの形状を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **Marker.Dot**.

継承元 **SeriesBase**
型 **Marker**

- symbolSize

この**Series** をレンダリングするために使用されるシンボルのサイズ（ピクセル単位）を取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

The default value for this property is **10** pixels.

継承元 **SeriesBase**
型 **number**

- symbolStyle

系列のシンボルスタイルを取得または設定します。 Scatter、LineSymbols、およびSplineSymbolsチャートタイプに適用されます。

継承元 **SeriesBase**
型 **any**

- visibility

系列を表示するかどうか、表示する場合はどこに表示するかを示す列挙値を取得または設定します。 The default value for this property is **SeriesVisibility.Visible**.

継承元 **SeriesBase**
型 **SeriesVisibility**

wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'series'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

dataToPoint

```
dataToPoint(pt: Point): Point
```

Point を系列データ座標からコントロール座標に変換します。

パラメーター

- **pt**: **Point**
  系列データ座標の**Point** 。

継承元 **SeriesBase**
戻り値 **Point**

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

指定した位置に凡例項目を描画します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。
- **rect**: **Rect**
  凡例項目の位置。
- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **void**

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

データ座標の境界四角形を返します。

getDataRect()がnullを返すと、データ値に基づいて自動的に境界が計算されます。

パラメーター

- **currentRect**: **Rect** OPTIONAL
  このパラメータはオプションです。 このパラメータはオプションです。

- **calculatedRect**: **Rect** OPTIONAL
  チャートの計算された四角形。 このパラメータはオプションです。

継承元 **SeriesBase**
戻り値 **Rect**

getPlotElement

```
getPlotElement(pointIndex: number): any
```

指定したポイントインデックスに対応するプロット要素を取得します。

パラメーター

- **pointIndex**: **number**
  データポイントのインデックス。

継承元 **SeriesBase**
戻り値 **any**

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

指定したポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）。

- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **SeriesBase**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーして系列を初期化します。

パラメーター

- **options**: **any**
  系列の初期化データを含むJavaScriptオブジェクト。

継承元 **SeriesBase**
戻り値 **void**

legendItemLength

```
legendItemLength(): number
```

凡例にある系列項目の数を返します。

継承元 **SeriesBase**
戻り値 **number**

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

凡例項目の高さと幅を測定します。

パラメーター

- **engine**: **IRenderEngine**
  使用するレンダリングエンジン。

- **index**: **number**
  凡例項目のインデックス（系列が複数の凡例項目を持つ場合）。

継承元 **SeriesBase**
戻り値 **Size**

onRendered

```
onRendered(engine: IRenderEngine): void
```

rendered イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。

継承元 **SeriesBase**
戻り値 **void**

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

rendering イベントを発生させます。

パラメーター

- **engine**: **IRenderEngine**
  系列のレンダリングに使用される**IRenderEngine** オブジェクト。
- **index**: **number**
  レンダリングする系列のインデックス。
- **count**: **number**
  レンダリングする系列の合計数。

継承元 **SeriesBase**
戻り値 **boolean**

pointToData

```
pointToData(pt: Point): Point
```

Point をコントロール座標から系列データ座標に変換します。

パラメーター

- **pt**: **Point**
  変換するポイント（コントロール座標単位）。

継承元 **SeriesBase**
戻り値 **Point**

イベント

rendered

系列がレンダリングされたときに発生します。

継承元 **SeriesBase**
引数 **IRenderEngine**

rendering

系列がレンダリングされるときに発生します。

継承元 **SeriesBase**
引数 **EventArgs**

wijmo/wijmo.angular2.gauge モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.gauge**

wijmo.gaugeモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.gaugeは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjGauge from 'wijmo/wijmo.angular2.gauge';

@Component({
    directives: [wjGauge.WjLinearGauge],
    template: '<wj-linear-gauge [(value)]="amount" [isReadOnly]="false"></wj-linear-gauge>',
    selector: 'my-cmp',
})
export class MyCmp {
    amount = 0;
}
```

クラス

- WjBulletGraph
- WjLinearGauge
- WjRadialGauge
- WjRange

WjBulletGraph クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.gauge**
基本クラス **BulletGraph**
表示 ☐継承されたメンバー ☐イベント発生元

BulletGraph コントロールに対応するAngular 2コンポーネント。

wj-bullet-graphコンポーネントを使用して、Angular 2アプリケーションに**BulletGraph**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjBulletGraphコンポーネントは、**BulletGraph**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-bullet-graphコンポーネントには、**WjRange** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- bad
- controlTemplate
- direction
- face
- format
- getText
- good
- gotFocusNg
- hasShadow
- hostElement
- initialized
- isAnimated
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- lostFocusNg
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- stackRanges
- step
- target
- thickness

- thumbSize
- tickSpacing
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): BulletGraph
```

BulletGraph クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **BulletGraph**
戻り値 **BulletGraph**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

bad

指標が不良と見なされる基準値を取得または設定します。

継承元 **BulletGraph**
型 **number**

STATIC controlTemplate

Gauge コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **Gauge**
型 **any**

direction

ゲージの値が増加する方向を取得または設定します。 The default value for this property is **GaugeDirection.Right**.

継承元 **LinearGauge**
型 **GaugeDirection**

face

ゲージの全体的な形状と外観を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

format

ゲージ値をテキストとして表示するために使用される書式文字列を取得または設定します。

継承元 **Gauge**
型 **string**

getText

ゲージ値の表示に使用されるカスタマイズされた文字列を返す コールバックを取得または設定します。

このプロパティは、ゲージに表示される文字列をカスタマイズしたいが、 **format** プロパティでは十分でない場合に使用してください。

指定する場合、コールバック関数は、ゲージ、部分名、値、および 書式設定された値をパラメータとして取る必要があります。また、ゲージに表示される文字列を返す必要があります。

次に例を示します。

```
// 値を文字列に変換するコールバック
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return '空！';
      if (value <= 25) return '低量...';
      if (value <= 95) return '適量';
      return '満タン';
    case 'min':
      return '空';
    case 'max':
      return '満タン';
  }
  return text;
}
```

継承元 **Gauge**
型 **Function**

good

指標が良好と見なされる基準値を取得または設定します。

継承元 **BulletGraph**
型 **number**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

hasShadow

ゲージに影の効果を表示するかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isAnimated

Gauge が値の変更を示すためにアニメーションを使用するかどうかを決定する値を取得または設定します。 The default value for this property is **true**.

継承元 `Gauge`
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用して値を編集できるかどうかを示す値を 取得または設定します。

The default value for this property is **true**.

継承元 `Gauge`
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- max

ゲージに表示できる最大値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- min

ゲージに表示できる最小値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- origin

範囲を描画するための始点を取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、値の範囲はゲージの最小値から始まります。最小値がゼロ未満の場合はゼロから始まります。

継承元 **Gauge**
型 **number**

- pointer

ゲージの現在の値を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

- ranges

このゲージの範囲のコレクションを取得します。

継承元 **Gauge**
型 **ObservableArray**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

showRanges

ranges プロパティに含まれる範囲をゲージに表示するかどうかを示す値を取得または設定します。

このプロパティがfalseに設定されている場合、**ranges** プロパティに含まれる範囲はゲージに表示されません。 代わりに、これらは、値の変化のアニメーション中に、**pointer** 範囲の色を補間するために使用されます。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

showText

ゲージにテキストとして表示する**ShowText** の値を取得または設定します。 The default value for this property is **ShowText.All** for **RadialGauge** controls, and to **ShowText.None** for **LinearGauge** controls.

継承元 **Gauge**
型 **ShowText**

showTicks

ゲージで、各**step** 値に目盛りマークを表示するかどうかを決定するプロパティを 取得または設定します。

目盛りマークは、CSSで**wj-gauge**クラス名と**wj-ticks**クラス名を 使用して書式設定できます。次に例を示します。

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

The default value for this property is **false**.

継承元 **Gauge**
型 **boolean**

stackRanges

ranges コレクションに含まれる範囲をゲージ内に積み重ねるかどうかを決定する値を取得または設定します。

stackRanges のデフォルト値はfalseであり、**ranges** コレクションの範囲はゲージのフェイスと同じ太さで表示されます。**stackRanges** をtrueに設定すると、範囲が狭くなり、並んで表示されます。

継承元 **Gauge**
型 **boolean**

step

ユーザーが方向キーを押すかマウスホイールを回したときに**value** プロパティを増減する量を取得または設定します。

継承元 **Gauge**
型 **number**

- target

指標の目標値を取得または設定します。

継承元 **BulletGraph**
型 **number**

- thickness

ゲージの太さを0~1のスケールで取得または設定します。

thicknessを1に設定すると、ゲージは最大限に太くなります。値が小さいほどゲージは細くなります。

継承元 **Gauge**
型 **number**

- thumbSize

ゲージの現在の値を示す要素のサイズ（ピクセル単位）を取得または設定します。

継承元 **Gauge**
型 **number**

- tickSpacing

目盛りマークの間隔を取得または設定します。

ゲージ上に目盛りを表示するようにするには、**showTicks** プロパティをtrueに設定します。デフォルトでは、目盛りマークの間隔は**step** プロパティによって定義されます。

tickSpacing プロパティを使用してデフォルトをオーバーライドし、 **step** プロパティの値と異なるスペーシングを設定できます。デフォルトの動作に戻すには、**tickSpacing** プロパティをnullに設定します。

継承元 **Gauge**
型 **number**

- value

ゲージに表示される値を取得または設定します。

継承元 **Gauge**
型 **number**

- valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): number
```

指定したポイントにあるゲージの値に対応する数値を取得します。

例:

```
// ユーザーがゲージをクリックしたときにそのポイントのヒットテストを行います。
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）、MouseEvent オブジェクト、またはポイントのX座標。
- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **Gauge**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Gauge**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コン トロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示します。

継承元 **Gauge**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

value プロパティの値が変更されたときに発生します。

継承元 **Gauge**
引数 **EventArgs**

WjLinearGauge クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.gauge**
基本クラス **LinearGauge**
表示 ☐継承されたメンバー ☐イベント発生元

LinearGauge コントロールに対応するAngular 2コンポーネント。

wj-linear-gaugeコンポーネントを使用して、Angular 2アプリケーションに**LinearGauge**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjLinearGaugeコンポーネントは、**LinearGauge**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-linear-gaugeコンポーネントには、**WjRange** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- controlTemplate
- direction
- face
- format
- getText
- gotFocusNg
- hasShadow
- hostElement
- initialized
- isAnimated
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- lostFocusNg
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- stackRanges
- step
- thickness
- thumbSize
- tickSpacing
- value

- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): LinearGauge
```

LinearGauge クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **LinearGauge**
戻り値 **LinearGauge**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

STATIC controlTemplate

Gauge コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **Gauge**
型 **any**

direction

ゲージの値が増加する方向を取得または設定します。 The default value for this property is **GaugeDirection.Right**.

継承元 **LinearGauge**
型 **GaugeDirection**

face

ゲージの全体的な形状と外観を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

format

ゲージ値をテキストとして表示するために使用される書式文字列を取得または設定します。

継承元 **Gauge**
型 **string**

getText

ゲージ値の表示に使用されるカスタマイズされた文字列を返す コールバックを取得または設定します。

このプロパティは、ゲージに表示される文字列をカスタマイズしたいが、 **format** プロパティでは十分でない場合に使用してください。

指定する場合、コールバック関数は、ゲージ、部分名、値、および 書式設定された値をパラメータとして取る必要があります。また、ゲージに表示される文字列を返す必要があります。

次に例を示します。

```
// 値を文字列に変換するコールバック
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return '空！';
      if (value <= 25) return '低量...';
      if (value <= 95) return '適量';
      return '満タン';
    case 'min':
      return '空';
    case 'max':
      return '満タン';
  }
  return text;
}
```

継承元 **Gauge**
型 **Function**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

hasShadow

ゲージに影の効果を表示するかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isAnimated

Gauge が値の変更を示すためにアニメーションを使用するかどうかを決定する値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isReadOnly

ユーザーがマウスとキーボードを使用して値を編集できるかどうかを示す値を 取得または設定します。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- max

ゲージに表示できる最大値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- min

ゲージに表示できる最小値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- origin

範囲を描画するための始点を取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、値の範囲はゲージの最小値から始まります。最小値がゼロ未満の場合はゼロから始まります。

継承元 **Gauge**
型 **number**

- pointer

ゲージの現在の値を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

- ranges

このゲージの範囲のコレクションを取得します。

継承元 **Gauge**
型 **ObservableArray**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

showRanges

ranges プロパティに含まれる範囲をゲージに表示するかどうかを示す値を取得または設定します。

このプロパティがfalseに設定されている場合、**ranges** プロパティに含まれる範囲はゲージに表示されません。 代わりに、これらは、値の変化のアニメーション中に、**pointer** 範囲の色を補間するために使用されます。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

showText

ゲージにテキストとして表示する**ShowText** の値を取得または設定します。 The default value for this property is **ShowText.All** for **RadialGauge** controls, and to **ShowText.None** for **LinearGauge** controls.

継承元 **Gauge**
型 **ShowText**

showTicks

ゲージで、各**step** 値に目盛りマークを表示するかどうかを決定するプロパティを 取得または設定します。

目盛りマークは、CSSで**wj-gauge**クラス名と**wj-ticks**クラス名を 使用して書式設定できます。次に例を示します。

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

The default value for this property is **false**.

継承元 **Gauge**
型 **boolean**

stackRanges

ranges コレクションに含まれる範囲をゲージ内に積み重ねるかどうかを決定する値を取得または設定します。

stackRanges のデフォルト値はfalseであり、**ranges** コレクションの範囲はゲージのフェイスと同じ太さで表示されます。**stackRanges** をtrueに設定すると、範囲が狭くなり、並んで表示されます。

継承元 **Gauge**
型 **boolean**

step

ユーザーが方向キーを押すかマウスホイールを回したときに**value** プロパティを増減する量を取得または設定します。

継承元 **Gauge**
型 **number**

- thickness

ゲージの太さを0~1のスケールで取得または設定します。

thicknessを1に設定すると、ゲージは最大限に太くなります。値が小さいほどゲージは細くなります。

継承元 **Gauge**
型 **number**

- thumbSize

ゲージの現在の値を示す要素のサイズ（ピクセル単位）を取得または設定します。

継承元 **Gauge**
型 **number**

- tickSpacing

目盛りマークの間隔を取得または設定します。

ゲージ上に目盛りを表示するようにするには、**`showTicks`** プロパティをtrueに設定します。デフォルトでは、目盛りマークの間隔は**`step`** プロパティによって定義されます。

`tickSpacing` プロパティを使用してデフォルトをオーバーライドし、 **`step`** プロパティの値と異なるスペーシングを設定できます。デフォルトの動作に戻すには、**`tickSpacing`** プロパティをnullに設定します。

継承元 **Gauge**
型 **number**

- value

ゲージに表示される値を取得または設定します。

継承元 **Gauge**
型 **number**

- valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): number
```

指定したポイントにあるゲージの値に対応する数値を取得します。

例:

```
// ユーザーがゲージをクリックしたときにそのポイントのヒットテストを行います。
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）、MouseEvent オブジェクト、またはポイントのX座標。
- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **Gauge**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Gauge**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コン トロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示します。

継承元 **Gauge**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

value プロパティの値が変更されたときに発生します。

継承元 **Gauge**
引数 **EventArgs**

WjRadialGauge クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.gauge**
基本クラス **RadialGauge**
表示 ☐継承されたメンバー ☐イベント発生元

RadialGauge コントロールに対応するAngular 2コンポーネント。

wj-radial-gaugeコンポーネントを使用して、Angular 2アプリケーションに**RadialGauge**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjRadialGaugeコンポーネントは、**RadialGauge**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

wj-radial-gaugeコンポーネントには、**WjRange** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoScale
- clientSize
- controlTemplate
- face
- format
- getText
- gotFocusNg
- hasShadow
- hostElement
- initialized
- isAnimated
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- lostFocusNg
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- stackRanges
- startAngle
- step
- sweepAngle
- thickness

- thumbSize
- tickSpacing
- value
- valueChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onValueChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- valueChanged

コンストラクタ

constructor

```
constructor(element: any, options?): RadialGauge
```

RadialGauge クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **RadialGauge**
戻り値 **RadialGauge**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoScale

ゲージがホスト要素に収まるように自動的に拡大縮小されるかどうかを示す値を取得または設定します The default value for this property is **true**.

継承元 **RadialGauge**
型 **boolean**

clientSize

autoScale 、パディング、枠線、および余白を考慮して、ゲージのクライアント領域のサイズを取得します。

継承元 **RadialGauge**
型 **Size**

STATIC controlTemplate

Gauge コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **Gauge**
型 **any**

face

ゲージの全体的な形状と外観を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

format

ゲージ値をテキストとして表示するために使用される書式文字列を取得または設定します。

継承元 **Gauge**
型 **string**

getText

ゲージ値の表示に使用されるカスタマイズされた文字列を返す コールバックを取得または設定します。

このプロパティは、ゲージに表示される文字列をカスタマイズしたいが、 **format** プロパティでは十分でない場合に使用してください。

指定する場合、コールバック関数は、ゲージ、部分名、値、および 書式設定された値をパラメータとして取る必要があります。また、ゲージに表示される文字列を返す必要があります。

次に例を示します。

```
// 値を文字列に変換するコールバック
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return '空！';
      if (value <= 25) return '低量...';
      if (value <= 95) return '適量';
      return '満タン';
    case 'min':
      return '空';
    case 'max':
      return '満タン';
  }
  return text;
}
```

継承元 **Gauge**
型 **Function**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

hasShadow

ゲージに影の効果を表示するかどうかを示す値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isAnimated

Gauge が値の変更を示すためにアニメーションを使用するかどうかを決定する値を取得または設定します。 The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isReadOnly

ユーザーがマウスとキーボードを使用して値を編集できるかどうかを示す値を 取得または設定します。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- max

ゲージに表示できる最大値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- min

ゲージに表示できる最小値を取得または設定します。

min および**max** プロパティの使用方法については、 「**minおよびmaxプロパティの使用**」トピックを参照してください。

継承元 **Gauge**
型 **number**

- origin

範囲を描画するための始点を取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、値の範囲はゲージの最小値から始まります。最小値がゼロ未満の場合はゼロから始まります。

継承元 **Gauge**
型 **number**

- pointer

ゲージの現在の値を表すために使用される**Range** を取得または設定します。

継承元 **Gauge**
型 **Range**

- ranges

このゲージの範囲のコレクションを取得します。

継承元 **Gauge**
型 **ObservableArray**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

showRanges

ranges プロパティに含まれる範囲をゲージに表示するかどうかを示す値を取得または設定します。

このプロパティがfalseに設定されている場合、**ranges** プロパティに含まれる範囲はゲージに表示されません。 代わりに、これらは、値の変化のアニメーション中に、**pointer** 範囲の色を補間するために使用されます。

The default value for this property is **true**.

継承元 **Gauge**
型 **boolean**

showText

ゲージにテキストとして表示する**ShowText** の値を取得または設定します。 The default value for this property is **ShowText.All** for **RadialGauge** controls, and to **ShowText.None** for **LinearGauge** controls.

継承元 **Gauge**
型 **ShowText**

showTicks

ゲージで、各**step** 値に目盛りマークを表示するかどうかを決定するプロパティを 取得または設定します。

目盛りマークは、CSSで**wj-gauge**クラス名と**wj-ticks**クラス名を 使用して書式設定できます。次に例を示します。

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

The default value for this property is **false**.

継承元 **Gauge**
型 **boolean**

stackRanges

ranges コレクションに含まれる範囲をゲージ内に積み重ねるかどうかを決定する値を取得または設定します。

stackRanges のデフォルト値はfalseであり、**ranges** コレクションの範囲はゲージのフェイスと同じ太さで表示されます。**stackRanges** をtrueに設定すると、範囲が狭くなり、並んで表示されます。

継承元 **Gauge**
型 **boolean**

startAngle

ゲージの開始角度（度単位）を取得または設定します。

角度は9時の位置から時計回りに度単位で測定されます。

The default value for this property is **0**.

継承元 **RadialGauge**
型 **number**

step

ユーザーが方向キーを押すかマウスホイールを回したときに**value** プロパティを増減する量を取得または設定します。

継承元 **Gauge**
型 **number**

sweepAngle

ゲージの掃引角度（度単位）を取得または設定します。

角度は9時の位置から時計回りに度単位で測定されます。

The default value for this property is **180**.

継承元 **RadialGauge**
型 **number**

thickness

ゲージの太さを0~1のスケールで取得または設定します。

thicknessを1に設定すると、ゲージは最大限に太くなります。値が小さいほどゲージは細くなります。

継承元 **Gauge**
型 **number**

thumbSize

ゲージの現在の値を示す要素のサイズ（ピクセル単位）を取得または設定します。

継承元 **Gauge**
型 **number**

tickSpacing

目盛りマークの間隔を取得または設定します。

ゲージ上に目盛りを表示するようにするには、**showTicks** プロパティをtrueに設定します。デフォルトでは、目盛りマークの間隔は**step** プロパティによって定義されます。

tickSpacing プロパティを使用してデフォルトをオーバーライドし、 **step** プロパティの値と異なるスペーシングを設定できます。デフォルトの動作に戻すには、**tickSpacing** プロパティをnullに設定します。

継承元 **Gauge**
型 **number**

value

ゲージに表示される値を取得または設定します。

継承元 **Gauge**
型 **number**

valueChangedNg

プログラムによるアクセスに使用されるWijmo **valueChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **valueChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は'value'です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: number): number
```

指定したポイントにあるゲージの値に対応する数値を取得します。

例:

```
// ユーザーがゲージをクリックしたときにそのポイントのヒットテストを行います。
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

パラメーター

- **pt**: **any**
  調べるポイント（ウィンドウ座標単位）、MouseEvent オブジェクト、またはポイントのX座標。
- **y**: **number** OPTIONAL
  ポイントのY座標（最初のパラメーターが数値の場合）。

継承元 **Gauge**
戻り値 **number**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

valueChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Gauge**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示します。

継承元 **RadialGauge**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

valueChanged

value プロパティの値が変更されたときに発生します。

継承元 **Gauge**
引数 **EventArgs**

WjRange クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.gauge**
基本クラス **Range**
表示 ☐継承されたメンバー ☐イベント発生元

Range コントロールに対応するAngular 2コンポーネント。

wj-rangeコンポーネントは、 **WjLinearGauge**、 **WjRadialGauge**、 または**WjBulletGraph** コンポーネントに含める必要があります。

wj-rangeコンポーネントを使用して、Angular 2アプリケーションに**Range**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjRangeコンポーネントは、**Range**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- color
- initialized
- isInitialized
- max
- min
- name
- thickness
- wjProperty

メソッド

- created
- onPropertyChanged

イベント

- propertyChanged

コンストラクタ

constructor

```
constructor(options?: string): Range
```

Range クラスの新しいインスタンスを初期化します。

パラメーター

- **options**: **string** OPTIONAL
  Range 、または**Range** の名前を含む文字列の初期化オプション。

継承元 **Range**
戻り値 **Range**

プロパティ

- color

この範囲の表示に使用される色を取得または設定します。

継承元 **Range**
型 **string**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- max

この範囲の最大値を取得または設定します。

継承元 **Range**
型 **number**

- min

この範囲の最小値を取得または設定します。

継承元 **Range**
型 **number**

- name

この**Range** の名前を取得または設定します。

継承元 **Range**
型 **string**

- thickness

この範囲の太さを親ゲージの太さの割合（%）として取得または設定します。

継承元 **Range**
型 **number**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'ranges'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

onPropertyChanged

```
onPropertyChanged(e: PropertyChangedEventArgs): void
```

propertyChanged イベントを発生させます。

パラメーター

- **e**: **PropertyChangedEventArgs**
  プロパティ名および新旧の値を含む**PropertyChangedEventArgs** 。

継承元 **Range**
戻り値 **void**

イベント

propertyChanged

このRange のプロパティ値が変更されると発生します。

継承元 **Range**
引数 **PropertyChangedEventArgs**

wijmo/wijmo.angular2.olap モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.olap**

wijmo.olapモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.olapは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjOlap from 'wijmo/wijmo.angular2.olap';

@Component({
    directives: [wjOlap.WjPivotGrid],
    template: '<wj-pivot-grid [itemsSource]="data"></wj-pivot-grid>',
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjPivotChart
- WjPivotGrid
- WjPivotPanel
- WjSlicer

WjPivotChart クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.olap**
基本クラス **PivotChart**
表示 ☐継承されたメンバー ☐イベント発生元

PivotChart コントロールに対応するAngular 2コンポーネント。

wj-pivot-chartコンポーネントを使用して、Angular 2アプリケーションに**PivotChart**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjPivotChartコンポーネントは、**PivotChart**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- chartType
- engine
- flexChart
- flexPie
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemsSource
- legendPosition
- lostFocusNg
- maxPoints
- maxSeries
- rightToLeft
- showHierarchicalAxes
- showLegend
- showTitle
- showTotals
- stacking
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created

- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, options?): PivotChart
```

PivotChart クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **PivotChart**
戻り値 **PivotChart**

プロパティ

- chartType

作成するチャートのタイプを取得または設定します。

The default value for this property is **PivotChartType.Column**.

継承元 **PivotChart**
型 **PivotChartType**

- engine

この**PivotChart** を所有する**PivotEngine** への参照を取得します。

継承元 **PivotChart**
型 **PivotEngine**

- flexChart

内部の**FlexChart**コントロールへの参照を取得します。

継承元 **PivotChart**
型 **FlexChart**

- flexPie

内部の**FlexPie**コントロールへの参照を取得します。

継承元 **PivotChart**
型 **FlexPie**

- footer

チャートのフッタに表示されるテキストを取得または設定します。

継承元 **PivotChart**
型 **string**

- footerStyle

チャートのフッタスタイルを取得または設定します。

継承元 **PivotChart**
型 **any**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

header

チャートのヘッダに表示されるテキストを取得または設定します。

継承元 `PivotChart`
型 **string**

headerStyle

チャートのヘッダスタイルを取得または設定します。

継承元 `PivotChart`
型 **any**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- itemsSource

この**PivotChart** のデータを提供する**PivotEngine** または**PivotPanel** を 取得または設定します。

継承元 `PivotChart`
型 **any**

- legendPosition

凡例を表示するかどうか、表示する場合はプロットエリアに対してどの位置に表示するかを決定する値を取得または設定します。

The default value for this property is **Position.Right**.

継承元 `PivotChart`
型 `Position`

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- maxPoints

各系列に表示するポイントの最大数を取得または設定します。

The default value for this property is **100** points.

継承元 `PivotChart`
型 **number**

- maxSeries

チャートに表示するデータ系列の最大数を取得または設定します。

The default value for this property is **100** series.

継承元 `PivotChart`
型 **number**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- showHierarchicalAxes

グループ化されたデータに対してチャートの軸注釈をグループ化するかどうかを決定する値を　取得または設定します。

The default value for this property is **true**.

継承元 **PivotChart**
型 **boolean**

- showLegend

チャートに凡例を含めるかどうかを決定する値を取得または設定します。

The default value for this property is **LegendVisibility.Always**.

継承元 **PivotChart**
型 **LegendVisibility**

- showTitle

チャートにタイトルを含めるかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **PivotChart**
型 **boolean**

- showTotals

チャートに合計だけを含めるかどうかを決定する値を取得または設定します。

If showTotals is true and the view has Column Fields, then the chart will show column totals instead of individual values.

The default value for this property is **false**.

継承元 **PivotChart**
型 **boolean**

- stacking

系列オブジェクトを積み重ねるかどうかを決定する値と、積み重ねる場合はその方法を取得または設定します。

The default value for this property is **Stacking.None**.

継承元 **PivotChart**
型 **Stacking**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 `Control`
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `PivotChart`
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 `Control`
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **format**: **ImageFormat**
  エクスポートされる画像の**ImageFormat** 。
- **done**: **Function**
  データURLの生成後に呼び出される関数。 この関数は、引数としてデータURLに渡されます。

継承元 **PivotChart**
戻り値 **void**

saveImageToFile

```
saveImageToFile(filename: string): void
```

チャートを画像ファイルに保存します。

このメソッドはIEブラウザでは機能しないことにご注意ください。IEのサポートが必要な場合は、 `wijmo.chart.render` モジュールをページに追加してください。

パラメーター

- **filename**: **string**
  拡張子を含む、エクスポートされる画像ファイルの名前。サポートされているタイプは、PNG、JPEG　およびSVGです。

継承元 **PivotChart**
戻り値 **void**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

WjPivotGrid クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.olap**
基本クラス **PivotGrid**
表示 ☐継承されたメンバー ☐イベント発生元

PivotGrid コントロールに対応するAngular 2コンポーネント。

wj-pivot-gridコンポーネントを使用して、Angular 2アプリケーションに**PivotGrid**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjPivotGridコンポーネントは、**PivotGrid**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoScroll
- autoSearch
- autoSizedColumnNg
- autoSizedRowNg
- autoSizeMode
- autoSizingColumnNg
- autoSizingRowNg
- beginningEditNg
- bottomLeftCells
- cellEditEndedNg
- cellEditEndingNg
- cellFactory
- cells
- centerHeadersVertically
- childItemsPath
- clientSize
- cloneFrozenCells
- collapsibleSubtotals
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate

- copiedNg
- copyingNg
- customContextMenu
- deferResizing
- deletedRowNg
- deletingRowNg
- detailDialog
- draggedColumnNg
- draggedRowNg
- draggingColumnNg
- draggingColumnOverNg
- draggingRowNg
- draggingRowOverNg
- editableCollectionView
- editRange
- engine
- formatItemNg
- frozenColumns
- frozenRows
- gotFocusNg
- groupCollapsedChangedNg
- groupCollapsedChangingNg
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- initialized
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceChangedNg
- itemValidator
- keyActionEnter
- keyActionTab
- loadedRowsNg
- loadingRowsNg
- lostFocusNg
- mergeManager
- newRowAtTop
- pastedCellNg
- pastedNg
- pastingCellNg
- pastingNg
- prepareCellForEditNg
- preserveOutlineState

- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- resizedColumnNg
- resizedRowNg
- resizingColumnNg
- resizingRowNg
- rightToLeft
- rowAddedNg
- rowEditEndedNg
- rowEditEndingNg
- rowEditStartedNg
- rowEditStartingNg
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollPositionChangedNg
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionChangedNg
- selectionChangingNg
- selectionMode
- showAlternatingRows
- showColumnFieldHeaders
- showDetailOnDoubleClick
- showDropDown
- showErrors
- showGroups
- showMarquee
- showRowFieldHeaders
- showRowFieldSort
- showSelectedHeaders
- showSort
- sortedColumnNg
- sortingColumnNg
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- updatedLayoutNg
- updatedViewNg
- updatingLayoutNg
- updatingViewNg
- validateEdits
- viewRange
- virtualizationThreshold

- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseColumnsToLevel
- collapseGroupsToLevel
- collapseRowsToLevel
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getDetail
- getDetailView
- getKeys
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow

- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onItemsSourceChanging
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onRefreshed
- onRefreshing
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- showDetail

- startEditing
- toggleDropDownList

イベント

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- itemsSourceChanging
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- refreshed
- refreshing
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged
- selectionChanging

- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout
- updatingView

コンストラクタ

constructor

```
constructor(element: any, options?): PivotGrid
```

PivotGrid クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **PivotGrid**
戻り値 **PivotGrid**

プロパティ

activeEditor

現在アクティブになっているセルエディタを表す**HTMLInputElement**を取得します。

継承元 **FlexGrid**
型 **HTMLInputElement**

allowAddNew

ユーザーがソースコレクションに項目を追加できるようにグリッドに新規行テンプレートを表示するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、新規行テンプレートは表示されません。

継承元 **FlexGrid**
型 **boolean**

allowDelete

ユーザーが［Delete］キーを押したときにグリッドの選択されている行を削除するかどうかを示す値を取得または設定します。

isReadOnly プロパティがtrueに設定されている場合、選択されている行は削除されません。

継承元 **FlexGrid**
型 **boolean**

allowDragging

ユーザーがマウスを使用して行、列、またはその両方をドラッグできるかどうかを決定する値を取得または設定します。

autoScroll プロパティがtrueに設定されている場合、ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされます。

グリッドでは、列のドラッグがデフォルトで有効になっています。

グリッドが連結モードでは、行をドラッグするには特別な 考慮が必要になります。

グリッドが連結モードでは、行をドラッグするとデータソースとの同期が失われます（たとえば行4は6行として参照されることがあります）。 これを回避するには、**draggedRow** イベントを処理し、データを新しい行の位置と同期させる必要があります。

また、**allowSorting** プロパティをfalseに設定する必要があります。そうしないと、行の順序がデータによって決定され、行をドラッグするのは無意味になります。

次のフィドルでは、連結グリッドでの行のドラッグを実行しています。 **Bound Row Dragging (http://jsfiddle.net/Wijmo5/kyg0qsda/)**.

継承元 **FlexGrid**
型 **AllowDragging**

allowMerging

グリッドのどの部分のセル結合を許可するかを取得または設定します。

継承元 **FlexGrid**
型 **AllowMerging**

allowResizing

ユーザーがマウスを使用して行、列、またはその両方をサイズ変更できるかどうかを決定する値を取得または設定します。

サイズ変更が可能な場合は、列ヘッダーセルの右端をドラッグして列を、行ヘッダーセルの下端をドラッグして行をサイズ変更できます。

ヘッダーセルの端をダブルクリックして、行および列をコンテンツに合わせて自動的にサイズ変更することもできます。 自動サイズ変更の動作は、**autoSizeMode** プロパティを使用してカスタマイズできます。

継承元 **FlexGrid**
型 **AllowResizing**

allowSorting

ユーザーが列ヘッダーセルをクリックして列をソートできるかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

autoClipboard

グリッドがクリップボードショートカットを処理するかどうかを決定する値を取得または設定します。

次のクリップボードショートカットがあります。

［Ctrl］+［C］、［Ctrl］+［Ins］	グリッドの選択範囲をクリップボードにコピーします。
［Ctrl］+［V］、［Shift］+［Ins］	クリップボードのテキストをグリッドの選択範囲に貼り付けます。

クリップボード操作は、表示されている行および列だけが対象となります。

読み取り専用のセルは、貼り付け操作の影響を受けません。

継承元 **FlexGrid**
型 **boolean**

autoGenerateColumns

グリッドが**itemsSource** に基づいて列を自動的に生成するかどうかを 決定する値を取得または設定します。

列の生成は、少なくとも1項目を含む**itemsSource** プロパティに依存します。このデータ項目が検査され、列が作成されて、プリミティブ値（数値、文字列、Boolean、またはDate）を含むプロパティに連結されます。

プロパティをnullに設定すると、適切なタイプを推測する方法がないため、列は生成されません。このような場合は、**autoGenerateColumns** プロパティをfalseに設定し、明示的に列を作成する必要があります。次に例を示します。

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // データ項目にnull値が含まれる場合があります
  columns: [                  // そのため、列を明示的に定義します
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

継承元 **FlexGrid**
型 **boolean**

autoScroll

ユーザーが行または列を新しい位置にドラッグするときにグリッドの内容が自動的にスクロールされるかどうかを決定する値を取得または設定します。

行と列のドラッグは、**allowDragging** プロパティによって制御されます。

このプロパティはデフォルトでtrueに設定されます。

継承元 **FlexGrid**
型 **boolean**

autoSearch

ユーザーが読み取り専用セルに入力するに伴ってグリッドでセルを検索するかどうかを決定する値を取得または設定します。

検索が現在選択されている列(編集不可能な場合)で行われます。検索は現在選択されている行から始まり、大文字と小文字を区別されません。

継承元 **FlexGrid**
型 **boolean**

autoSizedColumnNg

プログラムによるアクセスに使用されるWijmo **autoSizedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

autoSizedRowNg

プログラムによるアクセスに使用されるWijmo **autoSizedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

autoSizeMode

行または列のサイズを自動設定するときにどのセルを考慮に入れるかを取得または設定します。

このプロパティは、ユーザーが列ヘッダの端をダブルクリックしたときの動作を制御します。

デフォルトでは、その列のヘッダセルとデータセルの内容に基づいて列の幅が自動的に設定されます。このプロパティを使用すると、ヘッダのみまたはデータのみに基づいて列の幅を設定するように変更できます。

継承元 **FlexGrid**
型 **AutoSizeMode**

autoSizingColumnNg

プログラムによるアクセスに使用されるWijmo **autoSizingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

autoSizingRowNg

プログラムによるアクセスに使用されるWijmo **autoSizingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **autoSizingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

beginningEditNg

プログラムによるアクセスに使用されるWijmo **beginningEdit**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **beginningEdit** Wijmoイベント名を使用してください。

型 **EventEmitter**

bottomLeftCells

左下のセルを含む**GridPanel** を取得します。

bottomLeftCells パネルは、行ヘッダの下、**columnFooters** パネルの左に表示されます。

継承元 **FlexGrid**
型 **GridPanel**

cellEditEndedNg

プログラムによるアクセスに使用されるWijmo **cellEditEnded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **cellEditEnded** Wijmoイベント名を使用してください。

型 **EventEmitter**

cellEditEndingNg

プログラムによるアクセスに使用されるWijmo **cellEditEnding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **cellEditEnding** Wijmoイベント名を使用してください。

型 **EventEmitter**

cellFactory

このグリッドのセルを作成および更新する**CellFactory** を取得または設定します。

継承元 **FlexGrid**
型 **CellFactory**

cells

データセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

centerHeadersVertically

ヘッダーセルのコンテンツを垂直方向の中央揃えで配置するかどうかを決定する値を 取得または設定します。

The default value for this property is **true**.

継承元 **PivotGrid**
型 **boolean**

childItemsPath

階層グリッドで子の行の生成に使用される 1 つ以上のプロパティの名前を取得または設定します。

このプロパティには、項目の子項目を含むプロパティの名前を指定する文字列を設定します （たとえば、 `childItemsPath = 'items'` ）。

項目の異なるレベルに異なる名前を持つ子項目がある場合は、このプロパティに、各レベルの子項目を含むプロパティの名前から成る配列を設定します。（たとえば、 `childItemsPath = ['checks','earnings'];` ）。

サンプル

Show me (https://jsfiddle.net/Wijmo5/kk9j93bL)

継承元 **FlexGrid**
型 **any**

clientSize

コントロールのクライアントサイズを取得します（コントロールのサイズからヘッダーとスクロールバーを引いた値）。

継承元 **FlexGrid**
型 **Size**

cloneFrozenCells

スクロール中に知覚されるパフォーマンスを向上させるには、FlexGridがフリーズされたセルを複製し、別の要素に表示するかどうかを決定する値を取得または設定します。

このプロパティのデフォルト値はnullであり、ブラウザーによって一番適当な設定が選択されます。

継承元 **FlexGrid**
型 **boolean**

collapsibleSubtotals

ユーザーがグリッドで行および列の小計グループの折りたたみと展開を行えるかどうかを 決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **PivotGrid**
型 **boolean**

collectionView

グリッドデータを含む**ICollectionView** を取得します。

継承元 **FlexGrid**
型 **ICollectionView**

columnFooters

列フッターセルを保持する**GridPanel** を取得します。

columnFooters パネルは、グリッドセルの下、**bottomLeftCells** パネルの右に表示されます。これを使用して、グリッドデータの下にサマリー情報を表示できます。

以下の例では、**columnFooters** パネルに 1行を追加して、**aggregate** プロパティが設定された列に サマリーデータを表示する方法を示します。

```
function addFooterRow(flex) {

  // 集計を表示するGroupRowを作成します
  var row = new wijmo.grid.GroupRow();

  // 列フッターパネルに行を追加します
  flex.columnFooters.rows.push(row);

  // ヘッダーにシグマを表示します
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

継承元 **FlexGrid**
型 **GridPanel**

columnHeaders

列ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

columnLayout

現在の列レイアウトを定義するJSON文字列を取得または設定します。

列レイアウト文字列は、列とそのプロパティを含む配列を表します。これを使用して、ユーザーが定義した列レイアウトをセッションを越えて保持できます。また、ユーザーが列レイアウトを変更できるアプリケーションで元に戻す/やり直し機能を実装することもできます。

データマップはシリアル化できないため、列レイアウト文字列に**dataMap** プロパティは含まれていません。

継承元 **FlexGrid**
型 **string**

columns

グリッドの列コレクションを取得します。

継承元 **FlexGrid**
型 **ColumnCollection**

controlRect

コントロールの外接矩形（ページ座標単位）を取得します。

継承元 **FlexGrid**
型 **Rect**

STATIC controlTemplate

FlexGrid コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **FlexGrid**
型 **any**

copiedNg

プログラムによるアクセスに使用されるWijmo **copied**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **copied** Wijmoイベント名を使用してください。

型 **EventEmitter**

copyingNg

プログラムによるアクセスに使用されるWijmo **copying**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **copying** Wijmoイベント名を使用してください。

型 **EventEmitter**

customContextMenu

グリッドがカスタムコンテキストメニューを提供するかどうかを決定する値を取得または設定します。

カスタムコンテキストメニューには、フィールド設定の変更、フィールドの削除、 グリッドセルの詳細レコードの表示などを行うためのコマンドが含まれます。

The default value for this property is **true**.

継承元 **PivotGrid**
型 **boolean**

deferResizing

ユーザーがマウスボタンを放すまで、行および列のサイズ変更を遅らせるかどうかを決定する値を取得または設定します。

デフォルトでは、**deferResizing** はfalseに設定されており、ユーザーがマウスをドラッグするに伴って行および列がサイズ変更されます。このプロパティをtrueに設定すると、グリッドにサイズ変更マーカーが表示され、ユーザーがマウスボタンを放したときに初めて行または列のサイズが変更されます。

継承元 **FlexGrid**
型 **boolean**

deletedRowNg

プログラムによるアクセスに使用されるWijmo **deletedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **deletedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

deletingRowNg

プログラムによるアクセスに使用されるWijmo **deletingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **deletingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

detailDialog

Gets a reference to the **DetailDialog** used to display the detail records when the user double-clicks a cell.

This property can be used to customize the content of the **DetailDialog**.

See also the **showDetailOnDoubleClick** property and the **showDetail** method.

継承元 **PivotGrid**
型 **DetailDialog**

draggedColumnNg

プログラムによるアクセスに使用されるWijmo **draggedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggedRowNg

プログラムによるアクセスに使用されるWijmo **draggedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingColumnNg

プログラムによるアクセスに使用されるWijmo **draggingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingColumnOverNg

プログラムによるアクセスに使用されるWijmo **draggingColumnOver**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingColumnOver** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingRowNg

プログラムによるアクセスに使用されるWijmo **draggingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

draggingRowOverNg

プログラムによるアクセスに使用されるWijmo **draggingRowOver**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **draggingRowOver** Wijmoイベント名を使用してください。

型 **EventEmitter**

editableCollectionView

グリッドデータを含む**IEditableCollectionView** を取得します。

継承元 **FlexGrid**
型 **IEditableCollectionView**

editRange

現在編集中のセルを識別する**CellRange** を取得します。

継承元 **FlexGrid**
型 **CellRange**

engine

この**PivotGrid** を所有する**PivotEngine** への参照を取得します。

継承元 **PivotGrid**
型 **PivotEngine**

formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

frozenColumns

固定された列の数を取得または設定します。

固定された列は水平方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

frozenRows

静止行の数を取得または設定します。

固定された行は垂直方向にスクロールできませんが、セルは選択および編集できます。

継承元 **FlexGrid**
型 **number**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupCollapsedChangedNg

プログラムによるアクセスに使用されるWijmo **groupCollapsedChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **groupCollapsedChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupCollapsedChangingNg

プログラムによるアクセスに使用されるWijmo **groupCollapsedChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **groupCollapsedChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

groupHeaderFormat

グループヘッダーコンテンツを作成するために使用される書式文字列を取得または設定します。

この文字列は、任意のテキストと次の置換文字列を含むことができます。

- **{name}**：グループ化されるプロパティの名前。
- **{value}**：グループ化されるプロパティの値。
- **{level}**：グループレベル。
- **{count}**：このグループ内にある項目の合計数。

列がグループ化プロパティに連結されている場合は、列ヘッダーを使用して パラメータを置換し、列の書式とデータマップを使用して `{value}` パラメータを計算します。 列がない場合は、代わりにグループ情報が使用されます。

グループプロパティに連結された非表示の列を追加して、グループヘッダーセルの書式設定をカスタマイズすることもできます。

このプロパティのデフォルト値は
`'{name}: <b>{value}</b>({count:n0} items)'` です。 これは、 `'Country: UK (12 items)'` や
`'Country: Japan (8 items)'` のようなグループヘッダーを作成します。

継承元 **FlexGrid**
型 **string**

headersVisibility

行ヘッダおよび列ヘッダが表示されるかどうかを決定する値を取得または設定します。

継承元 `FlexGrid`
型 `HeadersVisibility`

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 `HTMLElement`

imeEnabled

編集モードでないときに、グリッドがIME（Input Method Editor）をサポートするかどうかを決定する値を取得または設定します。

このプロパティは、日本語、中国語、韓国語など、IMEサポートを必要とする言語のサイト/アプリケーションにのみ関係します。

継承元 `FlexGrid`
型 `boolean`

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 `EventEmitter`

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 `boolean`

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 `boolean`

isReadOnly

ユーザーがマウスとキーボードを使用してセル値を変更できるかどうかを判定する値を取得または設定します。

継承元 `FlexGrid`
型 `boolean`

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemFormatter

このグリッドのセルのカスタマイズに使用されるフォーマッター関数を取得または設定します。

このフォーマッター関数は、任意のセルに任意の内容を追加できます。そのため、グリッドセルの外観と動作を非常に柔軟に制御することが可能です。

この関数は、セルを含む**GridPanel**、セルの行インデックスと列インデックス、セルを表すHTML要素の4つのパラメーターをとります。通常は、関数内でセル要素の**innerHTML** プロパティを変更するようにします。

例:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == wijmo.grid.CellType.Cell) {

    // セルにスパークラインを描画します。
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

FlexGridはセルをリサイクルするため、**itemFormatter** によってセルのスタイル属性を変更する場合は、 セルの適切でないスタイル属性を事前にリセットする必要があります。例:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // カスタマイズする属性をリセットします。
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // このセルのcolorおよびbackgroundColor属性をカスタマイズします。
  ...
}
```

複数のクライアントがグリッドのレンダリングをカスタマイズできるようにする場合は（たとえば、ディレクティブや再利用可能なライブラリを作成するときなど）、代わりに**formatItem** イベントを使用することを検討してください。このイベントを使用すると、複数のクライアントがそれぞれ独自のハンドラをアタッチできます。

継承元 **FlexGrid**
型 **Function**

itemsSource

グリッドに表示される項目を含む配列または**ICollectionView** を取得または設定します。

継承元 **FlexGrid**
型 **any**

itemsSourceChangedNg

プログラムによるアクセスに使用されるWijmo **itemsSourceChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemsSourceChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

itemValidator

セルに有効なデータが含まれているかどうかを決定するバリデータ関数を取得または設定します。

指定された場合、バリデータ関数はセルの行インデックスと列インデックスを含む2つのパラメータをとり、エラーの説明を含む文字列を返す必要があります。

このプロパティは、バインドされていないグリッドを処理する場合に特に便利です。バインドされたグリッドは、代わりに **getError** プロパティを使用して検証できます。

この例では、セルのすぐ上のセルと同じデータがセルに含まれないようにする方法を示します。

```
// 上のセルは、現在のセルと同じ値が含まれていないことを確認します
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'これは重複した値です...'
    }
  }
  return null; // エラーなし
}
```

継承元 **FlexGrid**
型 **Function**

keyActionEnter

［ENTER］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は**MoveDown** となり、コントロールがカーソルを次の行に移動します。 この動作は標準的なExcel式の動作です。

継承元 **FlexGrid**
型 **KeyAction**

keyActionTab

［Tab］キーが押されたときに実行されるアクションを取得または設定します。

このプロパティの既定の設定は、**None** となります。これにより、Tabキーが押されたときにブラウザ上で次または前のコントロールを選択できます。これは、ページのアクセシビリティを向上させるために推奨される設定になります。

以前のバージョンでは、デフォルトは**Cycle** に設定されていました。これにより、コントロールがカーソルを、グリッドを横切って下部に移動しました。これは標準的なExcelの動作ですが、アクセシビリティには適していません。

また、**CycleOut** の設定が存在します。これにより、カーソルがセルを通じて移動（**Cycle** ）してから、最後または最初のセルが選択されたときにページの次/前のコントロールに移動します 。

継承元 **FlexGrid**
型 **KeyAction**

loadedRowsNg

プログラムによるアクセスに使用されるWijmo **loadedRows**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadedRows** Wijmoイベント名を使用してください。

型 **EventEmitter**

loadingRowsNg

プログラムによるアクセスに使用されるWijmo **loadingRows**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadingRows** Wijmoイベント名を使用してください。

型 **EventEmitter**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

mergeManager

セルの結合方法を決定する**MergeManager** オブジェクトを取得または設定します。

継承元 **FlexGrid**
型 **MergeManager**

newRowAtTop

新しい行テンプレートをグリッドの上部に配置するか、下部に配置するかを示す値を取得または設定します。

newRowAtTop プロパティをtrueに設定した場合、新しい行テンプレートを常に表示したままにする場合は、**frozenRows** プロパティを 1に設定します。これにより、新しい行テンプレートは上部に固定され、ビューからスクロールオフされなくなります。

allowAddNew プロパティがtrueに設定されている場合、および**itemsSource** オブジェクトが新しい項目の追加をサポートしている場合にのみ、新しい行テンプレートが表示されます。

継承元 **FlexGrid**
型 **boolean**

pastedCellNg

プログラムによるアクセスに使用されるWijmo **pastedCell**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pastedCell** Wijmoイベント名を使用してください。

型 **EventEmitter**

- pastedNg

プログラムによるアクセスに使用されるWijmo **pasted**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pasted** Wijmoイベント名を使用してください。

型 **EventEmitter**

- pastingCellNg

プログラムによるアクセスに使用されるWijmo **pastingCell**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pastingCell** Wijmoイベント名を使用してください。

型 **EventEmitter**

- pastingNg

プログラムによるアクセスに使用されるWijmo **pasting**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pasting** Wijmoイベント名を使用してください。

型 **EventEmitter**

- prepareCellForEditNg

プログラムによるアクセスに使用されるWijmo **prepareCellForEdit**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **prepareCellForEdit** Wijmoイベント名を使用してください。

型 **EventEmitter**

- preserveOutlineState

データがリフレッシュされたときに、グリッドがノードの展開/折りたたみ状態を保持するかどうかを決定する値を取得または設定します。

preserveOutlineState プロパティの実装は、JavaScriptの**Map** オブジェクトに基づいています。このオブジェクトはIE 9およびIE 10で利用できません。

継承元 **FlexGrid**
型 **boolean**

- preserveSelectedState

データがリフレッシュされたときに、グリッドが行の選択状態を保持するかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

- quickAutoSize

グリッドが列のサイズを自動調整するときに精度よりもパフォーマンスを最適化するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、自動サイズ変更の機能が無効になります。このプロパティをtrueに設定すると、各列の**quickAutoSize** プロパティの値に従って、その機能が有効になります。null（デフォルト値）に設定した場合、カスタムの**itemFormatter** を持たないグリッドの機能、または**itemFormatter** イベントにアタッチされたハンドラの機能が有効になります。

継承元 **FlexGrid**
型 **boolean**

resizedColumnNg

プログラムによるアクセスに使用されるWijmo **resizedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizedRowNg

プログラムによるアクセスに使用されるWijmo **resizedRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizedRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizingColumnNg

プログラムによるアクセスに使用されるWijmo **resizingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

resizingRowNg

プログラムによるアクセスに使用されるWijmo **resizingRow**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **resizingRow** Wijmoイベント名を使用してください。

型 **EventEmitter**

rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

rowAddedNg

プログラムによるアクセスに使用されるWijmo **rowAdded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowAdded** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditEndedNg

プログラムによるアクセスに使用されるWijmo **rowEditEnded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditEnded** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditEndingNg

プログラムによるアクセスに使用されるWijmo **rowEditEnding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditEnding** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditStartedNg

プログラムによるアクセスに使用されるWijmo **rowEditStarted**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditStarted** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowEditStartingNg

プログラムによるアクセスに使用されるWijmo **rowEditStarting**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **rowEditStarting** Wijmoイベント名を使用してください。

型 **EventEmitter**

rowHeaderPath

行ヘッダーセルを作成するために使用されるプロパティの名前を取得または設定します。

行ヘッダーセルは表示されないし、選択することもできません。アクセシビリティツールと組み合わせて使用することを意図しています。

継承元 **FlexGrid**
型 **string**

rowHeaders

行ヘッダセルを含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

rows

グリッドの行コレクションを取得します。

継承元 **FlexGrid**
型 **RowCollection**

scrollPosition

グリッドのスクロールバーの値を表す**Point** を取得または設定します。

継承元 **FlexGrid**
型 **Point**

scrollPositionChangedNg

プログラムによるアクセスに使用されるWijmo **scrollPositionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **scrollPositionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

scrollSize

グリッド内容のサイズ（ピクセル単位）を取得します。

継承元 **FlexGrid**
型 **Size**

selectedItems

現在選択されているデータ項目を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは **selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selectedRows

現在選択されている行を含む配列を取得または設定します。

メモ: このプロパティはすべての選択モードで取得できますが、設定できるのは **selectionMode** が**SelectionMode.ListBox** に設定されているときだけです。

継承元 **FlexGrid**
型 **any**[]

selection

現在の選択を取得または設定します。

継承元 **FlexGrid**
型 **CellRange**

selectionChangedNg

プログラムによるアクセスに使用されるWijmo **selectionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionChangingNg

プログラムによるアクセスに使用されるWijmo **selectionChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectionChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectionMode

現在の選択モードを取得または設定します。

継承元 **FlexGrid**
型 **SelectionMode**

showAlternatingRows

グリッドで交互表示行のセルに'wj-alt'クラスを追加するかどうかを決定する値を取得または設定します。

このプロパティをfalseに設定すると、CSSを変更しなくても、交互表示行スタイルを無効にできます。

継承元 **FlexGrid**
型 **boolean**

showColumnFieldHeaders

グリッドで、左上のパネルに列フィールドヘッダーを 表示するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **PivotGrid**
型 **boolean**

showDetailOnDoubleClick

ユーザーがセルをダブルクリックしたときに、グリッドが詳細レコードを含むポップアップを表示するかどうかを決定する値を取得または設定します。The default value for this property is **true**.

継承元 **PivotGrid**
型 **boolean**

showDropDown

グリッドで、**showDropDown** プロパティがtrueに設定されている列のセルにドロップダウンボタンを追加するかどうかを示す値を取得または設定します。

これらのドロップダウンボタンは、列に**dataMap** が設定され、編集可能である場合にのみ表示されます。ユーザーがドロップダウンボタンをクリックすると、セルの値を選択するために使用できるリストがグリッドに表示されます。

セルのドロップダウンを使用するには、wijmo.inputモジュールをロードしておく必要があります。

継承元 **FlexGrid**
型 **boolean**

showErrors

グリッドで、検証エラーがあるセルとエラーの説明を含むツールチップに'wj-state-invalid' クラスを追加するかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**itemsSource** の**itemValidator** または**getError** プロパティを使用して、検証エラーを検出します。

継承元 **FlexGrid**
型 **boolean**

showGroups

データグループを区切るためにグリッドにグループ行を挿入するかどうかを決定する値を 取得または設定します。

データグループを作成するには、グリッドの **itemsSource** として使用される **ICollectionView** オブジェクトの **groupDescriptions** プロパティを変更します。

継承元 **FlexGrid**
型 **boolean**

showMarquee

現在の選択範囲の周囲にマーキー要素を表示するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

showRowFieldHeaders

グリッドで、左上のパネルに行フィールドヘッダーを 表示するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **PivotGrid**
型 **boolean**

showRowFieldSort

グリッドで、行フィールドの列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

通常の列ヘッダーとは異なり、行フィールドは常に昇順または降順でソートされます。このプロパティをtrueに設定した場合は、 すべての行フィールドヘッダーに常にソートアイコンが表示されます。

The default value for this property is **false**.

継承元 **PivotGrid**
型 **boolean**

showSelectedHeaders

選択されているヘッダセルを示すためにクラス名を追加するかどうかを示す値を取得または設定します。

継承元 **FlexGrid**
型 **HeadersVisibility**

showSort

グリッドで、列ヘッダーにソートインジケータを 表示するかどうかを決定する値を取得または設定します。

ソートは、グリッドの **itemsSource** として使用される **ICollectionView** オブジェクトの **sortDescriptions** プロパティによって制御されます。

継承元 **FlexGrid**
型 **boolean**

sortedColumnNg

プログラムによるアクセスに使用されるWijmo **sortedColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **sortedColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

sortingColumnNg

プログラムによるアクセスに使用されるWijmo **sortingColumn**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **sortingColumn** Wijmoイベント名を使用してください。

型 **EventEmitter**

sortRowIndex

ソートインジケーターを表示する列ヘッダパネルの行のインデックスを取得または設定します。

デフォルトでは、このプロパティはnullに設定されており、 **columnHeaders** パネルの最後の行がソート行になります。

継承元 **FlexGrid**
型 **number**

stickyHeaders

ユーザーがドキュメントをスクロールしたときに、列ヘッダーを表示したままにするかどうかを決定する値を取得または設定します。

継承元 **FlexGrid**
型 **boolean**

topLeftCells

左上のセル（列ヘッダーの左）を含む**GridPanel** を取得します。

継承元 **FlexGrid**
型 **GridPanel**

treeIndent

異なるレベルの行グループをオフセットするインデントを取得または設定します。

継承元 **FlexGrid**
型 **number**

updatedLayoutNg

プログラムによるアクセスに使用されるWijmo **updatedLayout**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatedLayout** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatedViewNg

プログラムによるアクセスに使用されるWijmo **updatedView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatedView** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatingLayoutNg

プログラムによるアクセスに使用されるWijmo **updatingLayout**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatingLayout** Wijmoイベント名を使用してください。

型 **EventEmitter**

updatingViewNg

プログラムによるアクセスに使用されるWijmo **updatingView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatingView** Wijmoイベント名を使用してください。

型 **EventEmitter**

validateEdits

検証に失敗した編集をユーザーがコミットしようとしたときに、グリッドを編集モードのままにするかどうかを指定する値を取得または設定します。

グリッドは、グリッドの**`itemsSource`** の**`getError`** メソッドを呼び出して、検証エラーを検出します。

継承元 **`FlexGrid`**
型 **boolean**

viewRange

現在表示されているセルの範囲を取得します。

継承元 **`FlexGrid`**
型 **`CellRange`**

virtualizationThreshold

仮想化を有効にするために必要な最小行数、最小列数、またはその両方を取得または設定します。

このプロパティはデフォルトでゼロに設定されます。つまり、仮想化は常に有効ということです。これにより、バインディングパフォーマンスとメモリー必要量が向上しますが、スクロール時のパフォーマンス低下は犠牲になります。

グリッドの行数が少ない場合（約50〜100）、このプロパティを150などのやや高い値に設定することで、スクロールのパフォーマンスを向上させることができます。これにより、仮想化が無効になり連結処理が遅くなりますが、認識されるスクロールのパフォーマンスが向上する可能性があります。たとえば、以下のコードは、データソースに150を超える項目がある場合、グリッドがセルを仮想化します。

```
// 150を超える項目がある場合はグリッドを仮想化します
theGrid.virtualizationThreshold = 150;
```

このプロパティを200より大きい値に設定することは推奨されません。ロード処理の時間が長くなり、グリッドはすべての行のセルを作成しているときに数秒間フリーズするため、ページ上の要素数が多いためブラウザが遅くなります。

行と列に別々の仮想化しきい値を設定する場合は、**virtualizationThreshold** プロパティを2つの数値を含む配列に設定できます。この場合、最初の数値は行のしきい値として使用され、2番目の数値は列のしきい値として使用されます。たとえば、次のコードでは、グリッドは行を仮想化しますが、列を仮想化しません。

```
// 行（しきい値0）は仮想化しますが、列は仮想化しません（しきい値10,000）
theGrid.virtualizationThreshold = [0, 10000];
```

継承元 **FlexGrid**
型 **any**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 `Control`
戻り値 **HTMLElement**

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

列をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **c**: **number**
  サイズ変更する列のインデックス。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 `FlexGrid`
戻り値 **void**

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

列の範囲をその内容がちょうど収まるようにサイズ変更します。

測定される行の範囲は常に、現在表示されているすべての行 + 現在表示されていない最大2,000行です。グリッドに大量のデータが含まれている場合には（たとえば、50,000行など）、すべての行を測定すると非常に時間がかかる可能性があるため、すべての行は測定されません。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstColumn**: **number** OPTIONAL
  サイズ変更する最初の列のインデックス（デフォルトは最初の列）。
- **lastColumn**: **number** OPTIONAL
  サイズ変更する最後の列のインデックス（デフォルトは最後の列）。
- **header**: **boolean** OPTIONAL
  列インデックスの参照先が通常の列であるか、ヘッダ列であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

行をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **r**: **number**
  サイズ変更する行のインデックス。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の列であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

行の範囲をその内容がちょうど収まるようにサイズ変更します。

このメソッドは、グリッドが表示されている場合にのみ機能します。ホスト要素がDOMに追加されていない場合、またはグリッドの祖先要素が非表示の場合、グリッドはセルを測定することができませんので、列のサイズを自動的に変更することはできません。

パラメーター

- **firstRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックス。
- **lastRow**: **number** OPTIONAL
  サイズ変更する最初の行のインデックスす。
- **header**: **boolean** OPTIONAL
  行インデックスの参照先が通常の行であるか、ヘッダ行であるか。
- **extra**: **number** OPTIONAL
  追加の間隔（ピクセル単位）。

継承元 **FlexGrid**
戻り値 **void**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

canEditCell

```
canEditCell(r: number, c: number): void
```

指定されたセルが編集可能かどうかを示す値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。

継承元 **FlexGrid**
戻り値 **void**

collapseColumnsToLevel

```
collapseColumnsToLevel(level: number): void
```

すべての列を指定されたレベルまで折りたたみます。

パラメーター

- **level**: **number**
  表示する最大列レベル。 0は総計だけを表示することを意味します。 1は最上位グループだけを表示することを意味します。 大きなレベルを指定すると、すべての列が展開されます。

継承元 **PivotGrid**
戻り値 **void**

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

すべてのグループ行を指定したレベルに折りたたみます。

パラメーター

- **level**: **number**
  表示する最大のグループレベル。

継承元 **FlexGrid**
戻り値 **void**

collapseRowsToLevel

```
collapseRowsToLevel(level: number): void
```

すべての行を指定されたレベルまで折りたたみます。

パラメーター

- **level**: **number**
  表示する最大行レベル。 0は総計だけを表示することを意味します。 1は最上位グループだけを表示することを意味します。 大きなレベルを指定すると、すべての行が展開されます。

継承元 **PivotGrid**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

継承元 **FlexGrid**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **FlexGrid**
戻り値 **void**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

継承元 **FlexGrid**
戻り値 **boolean**

focus

```
focus(): void
```

グリッド全体をスクロールしてビューに入れることなくグリッドにフォーカスを設定するようにオーバーライドされます。

継承元 **FlexGrid**
戻り値 **void**

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

セル要素の範囲（ビューポート座標単位）を取得します。

このメソッドは、**cells** パネル内のセル（スクロール可能なデータセル）の範囲を返します。その他のパネルにあるセルの範囲を取得するには、該当する**GridPanel** オブジェクトの**getCellBoundingRect** メソッドを使用してください。

戻り値は、セルの位置とサイズ（ビューポート座標単位）を含む**Rect** オブジェクトです。 このビューポート座標は、**getBoundingClientRect** メソッドで使用されている座標と同じです。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。

- **c**: **number**
  セルを含む列のインデックス。

- **raw**: **boolean** OPTIONAL
  返される矩形の単位をビューポート座標ではなく生のパネル座標にするかどうか。

継承元 **FlexGrid**
戻り値 **Rect**

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

グリッドのスクロール可能領域内のセルに格納された値を取得します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **number**
  セルを含む列のインデックス。
- **formatted**: **boolean**
  値を表示用に書式設定するかどうか。

継承元 **FlexGrid**
戻り値 **any**

getClipString

```
getClipString(rng?: CellRange): string
```

CellRange の内容を、クリップボードへのコピーに適した文字列として取得します。

非表示の行および列はクリップボード文字列に含まれません。

パラメーター

- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **string**

getColumn

```
getColumn(name: string): Column
```

名前または連結に基づいて列を取得します。

このメソッドは、名前で列を検索します。指定された名前を持つ列が見つからない場合は、バインディングによって検索します。検索では大文字と小文字が区別されます。

パラメーター

- **name**: **string**
  検索する名前または連結。

継承元 **FlexGrid**
戻り値 **Column**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getDetail

```
getDetail(row: number, col: number): any[]
```

指定されたグリッドセルに基づいて集約されたレコードを含む配列を取得します。

パラメーター

- **row**: **number**
  セルを含む行のインデックス。

- **col**: **number**
  セルを含む列のインデックス。

継承元 **PivotGrid**
戻り値 **any**[]

getDetailView

```
getDetailView(row: number, col: number): ICollectionView
```

特定のグリッドセルに基づいて集約されたレコードを含む**ICollectionView** を取得します。

パラメーター

- **row**: **number**
  セルを含む行のインデックス。

- **col**: **number**
  セルを含む列のインデックス。

継承元 **PivotGrid**
戻り値 **ICollectionView**

getKeys

```
getKeys(row: number, col: number): any
```

特定のセルの集約に使用されるフィールドと値に関する情報を 含むオブジェクトを取得します。

詳細については、@PivotEngine.getKeys メソッドを参照してください。

パラメーター

- **row**: **number**
  セルを含む行のインデックス。

- **col**: **number**
  セルを含む列のインデックス。

継承元 **PivotGrid**
戻り値 **any**

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

GridPanel 内のセルの結合範囲を示す**CellRange** を取得します。

パラメーター

- **p**: **GridPanel**
  範囲を含む**GridPanel** 。

- **r**: **number**
  セルを含む行のインデックス。

- **c**: **number**
  セルを含む列のインデックス。

- **clip**: **boolean** OPTIONAL
  結合範囲をグリッドの現在のビュー範囲にクリップするかどうか。

継承元 **FlexGrid**
戻り値 **CellRange**

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

セルの選択状態を示す**SelectedState** 値を取得します。

パラメーター

- **r**: **number**
  検査するセルの行インデックス。

- **c**: **number**
  検査するセルの列インデックス。

継承元 **FlexGrid**
戻り値 **SelectedState**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、**ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

指定されたポイントに関する情報を含む**HitTestInfo** オブジェクトを取得します。

次に例を示します。

```
// ユーザーがグリッドをクリックしたときに、ポイントをヒットテストします
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

パラメーター

- **pt**: **any**
  調べる**Point** （ページ座標単位）、MouseEventオブジェクト、またはポイントのX座標。

- **y**: **any** OPTIONAL
  ポイントのY座標（ページ座標単位、最初のパラメーターが数値の場合）。

継承元 **FlexGrid**
戻り値 **HitTestInfo**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

指定したCellRangeがこのグリッドの行および列コレクションに対して有効かどうかをチェックします。

パラメーター

- **rng**: **CellRange**
  チェックする範囲。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

autoSizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

autoSizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

autoSizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

autoSizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

beginningEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

cellEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

cellEditEnding イベントを発生させます。

パラメーター

- **e**: **CellEditEndingEventArgs**
  イベントデータを含む**CellEditEndingEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

copied イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

copying イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

deletedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

deletingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

draggedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

draggedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

draggingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

draggingColumnOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

draggingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

draggingRowOver イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatItemEventArgs**
  イベントデータを含む**FormatItemEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

groupCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

groupCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onItemsSourceChanging

```
onItemsSourceChanging(e: CancelEventArgs): boolean
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

loadedRows イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

loadingRows イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

pasted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

pastedCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

pasting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

pastingCell イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

prepareCellForEdit イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

resizedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

resizedRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

resizingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

resizingRow イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

rowAdded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

rowEditEnded イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

rowEditEnding イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

rowEditStarted イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

rowEditStarting イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

scrollPositionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

selectionChanged イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

selectionChanging イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

sortedColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **void**

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

sortingColumn イベントを発生させます。

パラメーター

- **e**: **CellRangeEventArgs**
  イベントデータを含む**CellRangeEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

updatedLayout イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **FlexGrid**
戻り値 **void**

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

updatingLayout イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

updatingView イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs**
  イベントデータを含む**CancelEventArgs** 。

継承元 **FlexGrid**
戻り値 **boolean**

refresh

```
refresh(fullUpdate?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

継承元 **FlexGrid**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

グリッドの表示を更新します。

パラメーター

- **fullUpdate**: **boolean**
  グリッドのレイアウトと内容を更新するか、内容だけを更新するか。

- **recycle**: **boolean** OPTIONAL
  既存の要素を再利用するかどうか。

- **state**: **boolean** OPTIONAL
  既存の要素を保持してその状態を更新するかどうか。

継承元 `FlexGrid`
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この`Control` が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。

- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。

- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。

- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 `Control`
戻り値 **number**

scrollIntoView

```
scrollIntoView(r: number, c: number, refresh?: boolean): boolean
```

指定したセルが画面に入るようにグリッドをスクロールします。

負の行と列のインデックスは無視されるので、以下を呼び出すと、

```
grid.scrollIntoView(200, -1);
```

グリッドは200行目を表示するように垂直にスクロールしますが、水平にスクロールしません。

パラメーター

- **r**: **number**
  画面に入るようにスクロールする行のインデックス
- **c**: **number**
  画面に入るようにスクロールする列のインデックス。
- **refresh**: **boolean** OPTIONAL
  スクロールした新しい位置をすぐ表示するためにグリッドを更新する必要があるかどうかを決定するオプションのパラメータ。

継承元 **FlexGrid**
戻り値 **boolean**

select

```
select(rng: any, show?: any): void
```

セル範囲を選択し、オプションでそのセル範囲が画面に入るようにスクロールします。

select メソッドは、**CellRange** 、と新しい選択範囲を画面に入るようにスクロールするかどうかを示すオプションのブール型パラメータを渡すことで呼び出すことができます。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(new CellRange(1, 1), true);

// 選択範囲（1,1）~（2,4）を指定し、画面に入るようにスクロールしません。
grid.select(new CellRange(1, 1, 2, 4), false);
```

select メソッドを呼び出してインデックスまたは選択する行と列を渡すこともできます。この場合、選択範囲が画面に入るようにスクロールします。以下に例を示しています。

```
// セル1,1を選択して画面に入るようにスクロールします
grid.select(1, 1);
```

パラメーター

- **rng**: **any**
  選択する範囲。
- **show**: **any** OPTIONAL
  新しい選択範囲が画面に入るようにスクロールするかどうか。

継承元 **FlexGrid**
戻り値 **void**

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

グリッドのスクロール可能領域内のセルの値を設定します。

パラメーター

- **r**: **number**
  セルを含む行のインデックス。
- **c**: **any**
  セルを含む列のインデックス、名前、またはバインディング。
- **value**: **any**
  セルに格納する値。
- **coerce**: **boolean** OPTIONAL
  列のデータ型に合わせて値を自動的に変更するかどうか。
- **invalidate**: **boolean** OPTIONAL
  グリッドを無効にして変更を表示するかどうか。

継承元 **FlexGrid**
戻り値 **boolean**

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

文字列を行および列に解析し、その内容を特定の範囲に適用します。

非表示の行および列はスキップされます。

パラメーター

- **text**: **string**
  グリッドに解析する、タブと改行で区切られたテキスト。
- **rng**: **CellRange** OPTIONAL
  コピーする**CellRange** 。 省略した場合、現在の選択範囲が使用されます。

継承元 **FlexGrid**
戻り値 **void**

showDetail

```
showDetail(row: number, col: number): void
```

指定されたグリッドセルの詳細を含むダイアログを表示します。

パラメーター

- **row**: **number**
  セルを含む行のインデックス。
- **col**: **number**
  セルを含む列のインデックス。

継承元 **PivotGrid**
戻り値 **void**

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

指定されたセルの編集を開始します。

FlexGrid の編集は、Excelの編集によく似ています。 ［F2］キーを押すか、セルをダブルクリックすると、 グリッドが**完全編集** モードになります。 このモードでは、ユーザーが［Enter］、［Tab］、または［Esc］キーを押すか、マウスを使用して選択範囲を移動するまで、セルエディタはアクティブのままになります。完全編集モードでは、カーソルキーを押しても、グリッドの編集モードは終了しません。

テキストをセルに直接入力すると、グリッドは**クイック編集**モードになります。 このモードでは、ユーザーがEnter、Tab、Esc、またはいずれかの矢印キーを押すまで、セルエディタはアクティブなままになります。

通常、完全編集モードは既存の値を変更する際に使用します。 クイック編集モードは新しいデータをすばやく入力する際に使用します。

編集中に［F2］キーを押すことで、完全編集モードとクイック編集モードを切り替えることができます。

パラメーター

- **fullEdit**: **boolean** OPTIONAL
  ユーザーがカーソルキーを押したときも編集モードのままにするかどうか。 デフォルトはtrueです。
- **r**: **number** OPTIONAL
  編集する行のインデックス。 デフォルトは現在選択されている行です。
- **c**: **number** OPTIONAL
  編集する列のインデックス。 デフォルトは現在選択されている列です。
- **focus**: **boolean** OPTIONAL
  編集開始時に、エディタにフォーカスを与えるかどうか。 デフォルトはtrueです。

継承元 **FlexGrid**
戻り値 **boolean**

toggleDropDownList

```
toggleDropDownList(): void
```

現在選択されているセルに関連付けられたドロップダウンリストボックスを切り替えます。

このメソッドでは、セルが編集モードに入ったとき、またはユーザが特定のキーを押したときに、ドロップダウンリストを自動的に表示することができます。

たとえば、このコードでは、グリッドが編集モードに入るたびに、グリッドにドロップダウンリストが表示されます。

```
// グリッドが編集モードに入ると、ドロップダウンリストを表示する。
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

このコードでは、ユーザーがスペースバーを押した場合、グリッドが編集モードに入った後で、ドロップダウンリストが表示されます。

```
// ユーザーがスペースバーを押したときにドロップダウンリストを表示する。
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

継承元 **FlexGrid**
戻り値 **void**

イベント

autoSizedColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizedRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingColumn

ユーザーが列ヘッダセルの右端をダブルクリックして列のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

autoSizingRow

ユーザーが行ヘッダセルの下端をダブルクリックして行のサイズを自動設定する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

beginningEdit

セルが編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnded

セル編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

cellEditEnding

セル編集が終了するときに発生します。

このイベントを使用して検証を実行し、無効な編集を防ぐことができます。 たとえば、次のコードは、ユーザーが文字「a」を含まない値を入力できないようにします。このコードは、編集が適用される前に、編集前と編集後の値を取得する方法を示しています。

```
function cellEditEnding (sender, e) {

  // 編集前と編集後の値を取得します
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // newValに「a」が含まれていない場合は、編集をキャンセルします
  e.cancel = newVal.indexOf('a') < 0;
}
```

cancel パラメータをtrueに設定すると、編集された値が無視されて、グリッドでセルの元の値が維持されます。

また、**stayInEditMode** パラメータをtrueに設定すると、グリッドの編集モードが維持され、編集をコミットする前にユーザーが無効な値を修正できます。

継承元 **FlexGrid**
引数 **CellEditEndingEventArgs**

copied

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーした後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

copying

ユーザーがいずれかのクリップボードショートカットキーを押して選択されている内容をクリップボードにコピーするときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラでコピー操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletedRow

ユーザーが［Del］キーを押して行を削除した後に発生します （**allowDelete** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

deletingRow

ユーザーが［Delete］キーを押して選択されている行を削除するときに発生します（**allowDelete** プロパティを参照）。

イベントハンドラで行の削除をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedColumn

ユーザーが列のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggedRow

ユーザーが行のドラッグを完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumn

ユーザーが列のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingColumnOver

ユーザーが列を別の位置にドラッグするときに発生します。

ハンドラは、このイベントをキャンセルして、ユーザーが特定の位置に列をドロップしないようにすることができます。次に例を示します。

```
// ドラッグされている列を記憶します
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// 'sales'列がインデックス0にドラッグされないようにします
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRow

ユーザーが行のドラッグを開始するときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

draggingRowOver

ユーザーが行を別の位置にドラッグするときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

formatItem

セルを表す要素が作成されたときに発生します。

このイベントを使用してセルを表示用に書式設定できます。このイベントは、目的は **itemFormatter** プロパティと同じですが、複数の独立したハンドラを使用できる利点があります。

以下のサンプルコードは、グループ行のセルから'wj-wrap'クラスを削除します。

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

継承元 **FlexGrid**
引数 **FormatItemEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

groupCollapsedChanged

グループが展開または折りたたまれた後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

groupCollapsedChanging

グループが展開または折りたたまれる直前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

itemsSourceChanged

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

itemsSourceChanging

グリッドが新しい項目ソースにバインドされた後に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

loadedRows

グリッド行がデータソースの項目に連結された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

loadingRows

グリッド行がデータソースの項目に連結される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

pasted

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastedCell

ユーザーがクリップボードの内容をセルに貼り付けた後に発生します（**autoClipboard** プロパティを参照）。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pasting

ユーザーがいずれかのクリップボードショートカットキーを押してクリップボードの内容を貼り付けた時に発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

pastingCell

ユーザーがクリップボードの内容をセルに貼り付けるときに発生します（**autoClipboard** プロパティを参照）。

イベントハンドラで貼り付け操作をキャンセルできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

prepareCellForEdit

エディタセルが作成されたとき、それがアクティブになる前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

resizedColumn

ユーザーが列のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizedRow

ユーザーが行のサイズ変更を完了したときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingColumn

列がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

resizingRow

行がサイズ変更されるときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowAdded

ユーザーが新規行テンプレートを編集して新しい項目を作成したときに発生します（**allowAddNew** プロパティを参照）。

イベントハンドラで新しい項目の内容をカスタマイズしたり、新しい項目の作成をキャンセルしたりできます。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnded

行編集が確定またはキャンセルされたときに発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditEnding

行編集が終了するとき、変更が確定またはキャンセルされる前に発生します。

このイベントを **rowEditStarted** イベントと組み合わせて使用して、ディープ連結編集を元に戻す操作を実装できます。次に例を示します。

```
// 編集の開始時にディープ連結値を保存します
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.')> -1) { // ディープ連結
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// 編集がキャンセルされたときにディープ連結値を復元します
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // ユーザーによって編集がキャンセルされました
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarted

行が編集モードに入った後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

rowEditStarting

行が編集モードに入る前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

scrollPositionChanged

コントロールがスクロールされた後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

selectionChanged

選択が変更された後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

selectionChanging

選択が変更される前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortedColumn

ユーザーが列ヘッダをクリックしてソートを適用した後に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

sortingColumn

ユーザーが列ヘッダをクリックしてソートを適用する前に発生します。

継承元 **FlexGrid**
引数 **CellRangeEventArgs**

updatedLayout

グリッドで内部レイアウトが更新された後に発生します。

継承元 **FlexGrid**
引数 **EventArgs**

updatedView

グリッドが現在のビューを構成する要素の作成/更新を完了したときに発生します。

グリッドのビューは以下のような操作に応じて更新されます。

- グリッドまたはそのデータソースの更新
- 行または列の追加、削除、変更
- グリッドのサイズ変更またはスクロール
- 選択の変更

継承元 **FlexGrid**
引数 **EventArgs**

updatingLayout

グリッドで内部レイアウトが更新される前に発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

updatingView

現在のビューを構成する要素の作成/更新をグリッドが開始したときに発生します。

継承元 **FlexGrid**
引数 **CancelEventArgs**

WjPivotPanel クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.olap**
基本クラス **PivotPanel**
表示 ☐継承されたメンバー ☐イベント発生元

PivotPanel コントロールに対応するAngular 2コンポーネント。

wj-pivot-panelコンポーネントを使用して、Angular 2アプリケーションに**PivotPanel**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjPivotPanelコンポーネントは、**PivotPanel**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- autoGenerateFields
- collectionView
- columnFields
- controlTemplate
- engine
- fields
- filterFields
- gotFocusNg
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- isViewDefined
- itemsSource
- itemsSourceChangedNg
- lostFocusNg
- pivotView
- restrictDragging
- rightToLeft
- rowFields
- showFieldIcons
- updatedViewNg
- updatingViewNg
- valueFields
- viewDefinition
- viewDefinitionChangedNg
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate

- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onItemsSourceChanged
- onLostFocus
- onRefreshed
- onRefreshing
- onUpdatedView
- onUpdatingView
- onViewDefinitionChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- itemsSourceChanged
- lostFocus
- refreshed
- refreshing
- updatedView
- updatingView
- viewDefinitionChanged

コンストラクタ

constructor

```
constructor(element: any, options?): PivotPanel
```

PivotPanel クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **PivotPanel**
戻り値 **PivotPanel**

プロパティ

autoGenerateFields

エンジンが**itemsSource** に基づいて**fields** コレクションを自動的に設定するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **PivotPanel**
型 **boolean**

collectionView

生データを含む**ICollectionView** を取得します。

継承元 **PivotPanel**
型 **ICollectionView**

columnFields

出力テーブル内の列を定義するフィールドのリストを取得します。

継承元 **PivotPanel**
型 **PivotFieldCollection**

STATIC controlTemplate

PivotPanel コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **PivotPanel**
型 **any**

engine

この**PivotPanel** によって制御される**PivotEngine** を取得または設定します。

継承元 **PivotPanel**
型 **PivotEngine**

- fields

ビューの構築に使用できるフィールドのリストを取得します。

継承元 `PivotPanel`
型 `PivotFieldCollection`

- filterFields

出力テーブルの生成時に適用されるフィルタを定義するフィールドのリストを取得します。

継承元 `PivotPanel`
型 `PivotFieldCollection`

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isViewDefined

ピボットビューが現在定義されているかどうかを判定する値を取得します。

ピボットビューが定義されているのは、**valueFields** リストが空でなく、 かつ**rowFields** リストと**columnFields** リストのいずれかが空でない場合です。

継承元 **PivotPanel**
型 **boolean**

- itemsSource

生データを含む配列または**ICollectionView** を取得または設定します。

継承元 **PivotPanel**
型 **any**

- itemsSourceChangedNg

プログラムによるアクセスに使用されるWijmo **itemsSourceChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemsSourceChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- pivotView

出力ピボットビューを含む**ICollectionView** を取得します。

継承元 **PivotPanel**
型 **ICollectionView**

- restrictDragging

パネルがフィールドのタイプに基づいてドラッグ操作を制限するかどうかを決定する値を取得または設定します。

このプロパティがtrueに設定されている場合、ディメンションフィールドを値フィールドリストにドラッグ、またメジャーフィールドを行または列のフィールドリストにドラッグすることができません。

このプロパティがfalseに設定されている場合、すべてのドラッグ操作が可能になります。

このプロパティがnull（デフォルト値）に設定されている場合、すべてのドラッグ操作が通常のデータソースに対して許可され、キューブデータソースでのドラッグが制限されます。

継承元 `PivotPanel`
型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- rowFields

出力テーブル内の行を定義するフィールドのリストを取得します。

継承元 `PivotPanel`
型 `PivotFieldCollection`

- showFieldIcons

メインフィールドリストに、フィールドがメジャーフィールドかディメンションフィールドかを示すアイコンを含めるかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 `PivotPanel`
型 **boolean**

- updatedViewNg

プログラムによるアクセスに使用されるWijmo **updatedView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatedView** Wijmoイベント名を使用してください。

型 **EventEmitter**

- updatingViewNg

プログラムによるアクセスに使用されるWijmo **updatingView**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **updatingView** Wijmoイベント名を使用してください。

型 **EventEmitter**

valueFields

出力テーブルに表示される値を定義するフィールドのリストを取得します。

継承元 **PivotPanel**
型 **PivotFieldCollection**

viewDefinition

現在のピボットビュー定義をJSON文字列として取得または設定します。

このプロパティは通常、現在のビューをアプリケーション設定として維持するために使用されます。

たとえば、次のコードは、ローカルストレージを使用してビュー定義を 保存およびロードする２つの関数を実装します。

```
// ビューを保存/ロードします
function saveView() {
  localStorage.viewDefinition = pivotPanel.viewDefinition;
}
function loadView() {
  pivotPanel.viewDefinition = localStorage.viewDefinition;
}
```

継承元 **PivotPanel**
型 **string**

viewDefinitionChangedNg

プログラムによるアクセスに使用されるWijmo **viewDefinitionChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **viewDefinitionChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は''です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **PivotPanel**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

updatedView イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **PivotPanel**
戻り値 **void**

onUpdatingView

```
onUpdatingView(e: ProgressEventArgs): void
```

updatingView イベントを発生させます。

パラメーター

- **e**: **ProgressEventArgs**
  イベントデータを提供する**ProgressEventArgs** 。

継承元 **PivotPanel**
戻り値 **void**

onViewDefinitionChanged

```
onViewDefinitionChanged(e?: EventArgs): void
```

viewDefinitionChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **PivotPanel**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化した後に発生します。

継承元 **PivotPanel**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

updatedView

エンジンが**pivotView** リストの更新を終了した後に発生します。

継承元 **PivotPanel**
引数 **EventArgs**

updatingView

エンジンが**pivotView** リストの更新を開始したときに発生します。

継承元 **PivotPanel**
引数 **ProgressEventArgs**

viewDefinitionChanged

ビューの定義が変更された後に発生します。

継承元 **PivotPanel**
引数 **EventArgs**

WjSlicer クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.olap**
基本クラス **Slicer**
表示 ☐継承されたメンバー ☐イベント発生元

Slicer コントロールに対応するAngular 2コンポーネント。

wj-slicerコンポーネントを使用して、Angular 2アプリケーションに**Slicer**コントロールを 追加できます。Angular 2マークアップの構文については、「Angular 2マークアップ構文」を参照してください。

WjSlicerコンポーネントは、**Slicer**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- controlTemplate
- field
- gotFocusNg
- header
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- lostFocusNg
- multiSelect
- rightToLeft
- showCheckboxes
- showHeader
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus

- onLostFocus
- onRefreshed
- onRefreshing
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing

コンストラクタ

constructor

```
constructor(element: any, options?): Slicer
```

Initializes a new instance of the **Slicer** class.

パラメーター

- **element**: **any**
  The DOM element that hosts the control, or a CSS selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

継承元 **Slicer**
戻り値 **Slicer**

プロパティ

- STATIC controlTemplate

Gets or sets the template used to instantiate **Slicer** controls.

継承元 **Slicer**
型 **any**

- field

Gets or sets the **PivotField** being filtered by this **Slicer**.

If the **PivotField** is not included in the current view definition, the **Slicer** will automatically add the field to the engine's **filterFields** collection.

If you want to remove the field from any **PivotPanel** controls so users cannot remove the field from the view definition, set the fields **visible** property to false. The field will remain active, but will not be shown in any **PivotPanel** controls.

継承元 **Slicer**
型 **PivotField**

- gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- header

Gets or sets the header string shown at the top of the **`Slicer`**.

The default value for this property is null, which causes the **`Slicer`** to display the **`field`** header at the top of the **`Slicer`**.

継承元 **`Slicer`**
型 **string**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **`Control`**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **`Control`**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **`Control`**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- multiSelect

Gets or sets a value that determines whether users should be allowed to select multiple values from the list.

The default value for this property is **false**.

継承元 `Slicer`
型 **boolean**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- showCheckboxes

Gets or sets a value indicating whether the control displays checkboxes next to each item.

The default value for this property is **false**.

継承元 `Slicer`
型 **boolean**

- showHeader

Gets or sets a value indicating whether the control displays the header area with the header string and multi-select/clear buttons.

The default value for this property is **true**.

継承元 `Slicer`
型 **boolean**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**`invalidateAll`** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

wijmo/wijmo.angular2.viewer モジュール

ファイル `wijmo.angular2.js`
モジュール **wijmo/wijmo.angular2.viewer**

wijmo.viewerモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.viewerは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjViewer from 'wijmo/wijmo.angular2.viewer';

@Component({
    directives: [wjViewer.WjReportViewer, wjViewer.WjPdfViewer],
    template: `
      <wj-report-viewer [reportName]="sales" [serviceUrl]="'webserviceApi'">
      </wj-report-viewer;`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjPdfViewer
- WjReportViewer

WjPdfViewer クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.viewer**
基本クラス **PdfViewer**
表示 ☐継承されたメンバー ☐イベント発生元

PdfViewer コントロールに対応するAngular 2コンポーネント。

wj-pdf-viewerコンポーネントを使用して、Angular 2アプリケーションに**PdfViewer**コントロールを追加します。 Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjPdfViewerコンポーネントは、**PdfViewer**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- beforeSendRequestNg
- controlTemplate
- filePath
- fullScreen
- fullScreenChangedNg
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- mouseMode
- mouseModeChangedNg
- pageIndex
- pageIndexChangedNg
- queryLoadingDataNg
- requestHeaders
- rightToLeft
- selectMouseMode
- selectMouseModeChangedNg
- serviceUrl
- thresholdWidth
- viewMode
- viewModeChangedNg
- wjModelProperty
- zoomFactor
- zoomFactorChangedNg
- zoomMode

メソッド

- addEventListener
- applyTemplate
- beginUpdate

- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- moveToPage
- onBeforeSendRequest
- onFullScreenChanged
- onGotFocus
- onLostFocus
- onMouseModeChanged
- onPageIndexChanged
- onQueryLoadingData
- onRefreshed
- onRefreshing
- onSelectMouseModeChanged
- onViewModeChanged
- onZoomFactorChanged
- refresh
- refreshAll
- reload
- removeEventListener
- showPageSetupDialog
- zoomToView
- zoomToViewWidth

イベント

- beforeSendRequest
- fullScreenChanged
- gotFocus
- lostFocus
- mouseModeChanged
- pageIndexChanged
- queryLoadingData
- refreshed
- refreshing
- selectMouseModeChanged
- viewModeChanged
- zoomFactorChanged

コンストラクタ

constructor

```
constructor(element: any, options?: any): PdfViewer
```

PdfViewer クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **PdfViewer**
戻り値 **PdfViewer**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

beforeSendRequestNg

プログラムによるアクセスに使用されるWijmo **beforeSendRequest**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **beforeSendRequest** Wijmoイベント名を使用してください。

型 **EventEmitter**

STATIC controlTemplate

ビューアコントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **ViewerBase**
型 **any**

filePath

サーバー上のドキュメントの完全パスを取得または設定します。

パスは、指定されたドキュメントをロードするためにサーバーに登録されているプロバイダのキーで始まります。

継承元 **ViewerBase**
型 **string**

fullScreen

ビューアが全画面表示モードかどうかを示す値を取得または設定します。

継承元 **ViewerBase**
型 **boolean**

- fullScreenChangedNg

プログラムによるアクセスに使用されるWijmo **fullScreenChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **fullScreenChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

mouseMode

マウスの動作を示す値を取得または設定します。

デフォルトはSelectToolです。その場合は、マウスでクリックしてドラッグすると、テキストが選択されます。

継承元 `ViewerBase`
型 `MouseMode`

mouseModeChangedNg

プログラムによるアクセスに使用されるWijmo **mouseModeChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **mouseModeChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

pageIndex

ビューパネルに現在表示されているページのインデックスを取得します。

継承元 `ViewerBase`
型 **number**

pageIndexChangedNg

プログラムによるアクセスに使用されるWijmo **pageIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pageIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

queryLoadingDataNg

プログラムによるアクセスに使用されるWijmo **queryLoadingData**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **queryLoadingData** Wijmoイベント名を使用してください。

型 **EventEmitter**

requestHeaders

データを送信または要求するときに使用する要求ヘッダーを含むオブジェクトを取得または設定します。

このプロパティは、認証が必要なシナリオでよく使用されます。以下に例を示しています。

```
viewer.requestHeaders = {
    Authorization: 'Bearer ' + appAuthService.getToken();
};
```

継承元 `ViewerBase`
型 **any**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- selectMouseMode

非推奨になりました。代わりにmouseModeを使用してください。

継承元 `ViewerBase`
型 **boolean**

- selectMouseModeChangedNg

プログラムによるアクセスに使用されるWijmo **selectMouseModeChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectMouseModeChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- serviceUrl

C1 Web APIサービスのアドレスを取得または設定します。

例："http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report"。

継承元 `ViewerBase`
型 **string**

- thresholdWidth

モバイルテンプレートとPCテンプレートの切り替えに使用するしきい値を取得または設定します。

デフォルト値は767pxです。コントロールの幅がthresholdWidthより小さい場合は、モバイルテンプレートが適用されます。コントロールの幅がthresholdWidth以上の場合は、PCテンプレートが適用されます。 thresholdWidthを0に設定すると、PCテンプレートだけが適用されます。 9999などの大きな数字に設定すると、モバイルテンプレートだけが適用されます。

継承元 `ViewerBase`
型 **number**

- viewMode

ドキュメントページの表示方法を示す値を取得または設定します。

継承元 `ViewerBase`
型 `ViewMode`

- viewModeChangedNg

プログラムによるアクセスに使用されるWijmo **viewModeChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **viewModeChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

- zoomFactor

ドキュメントページを表示する現在のズーム倍率を示す値を取得または設定します。

継承元 **ViewerBase**
型 **number**

- zoomFactorChangedNg

プログラムによるアクセスに使用されるWijmo **zoomFactorChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **zoomFactorChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- zoomMode

ドキュメントページを表示する現在のズームモードを示す値を取得または設定します。

継承元 **ViewerBase**
型 **ZoomMode**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

moveToPage

```
moveToPage(index: number): IPromise
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先のページの0から始まるインデックス。

継承元 **ViewerBase**
戻り値 **IPromise**

onBeforeSendRequest

```
onBeforeSendRequest(e: RequestEventArgs): void
```

beforeSendRequest イベントを発生させます。

パラメーター

- **e**: **RequestEventArgs**
  RequestEventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onFullScreenChanged

```
onFullScreenChanged(e?: EventArgs): void
```

fullScreenChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onMouseModeChanged

```
onMouseModeChanged(e?: EventArgs): void
```

mouseModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onPageIndexChanged

```
onPageIndexChanged(e?: EventArgs): void
```

pageIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onQueryLoadingData

```
onQueryLoadingData(e: QueryLoadingDataEventArgs): void
```

queryLoadingData イベントを発生させます。

パラメーター

- **e**: **QueryLoadingDataEventArgs**
  ロード中のデータを含む**QueryLoadingDataEventArgs** オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectMouseModeChanged

```
onSelectMouseModeChanged(e?: EventArgs): void
```

非推奨になりました。代わりにonMouseModeChangedを使用してください。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onViewModeChanged

```
onViewModeChanged(e?: EventArgs): void
```

viewModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onZoomFactorChanged

```
onZoomFactorChanged(e?: EventArgs): void
```

zoomFactorChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **ViewerBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

reload

```
reload(): void
```

ドキュメントを再ロードします。

これは、ドキュメントを強制的に再ロードおよび再レンダリングする場合に便利です。

継承元 **ViewerBase**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

showPageSetupDialog

```
showPageSetupDialog(): void
```

ページ設定ダイアログボックスを表示します。

継承元 **ViewerBase**
戻り値 **void**

zoomToView

```
zoomToView(): void
```

現在のページを拡大縮小して、ビューパネルにページ全体を表示します。

継承元 **ViewerBase**
戻り値 **void**

zoomToViewWidth

```
zoomToViewWidth(): void
```

ビューパネルの幅に合わせて現在のページを拡大縮小します。

継承元 **ViewerBase**
戻り値 **void**

イベント

beforeSendRequest

各要求がサーバーに送信される前に発生します。

このイベントでは、サーバーに送信する前に、URL、ヘッダー、データなどのリクエストオプションに加えてリクエストメソッドも変更できます。 このイベントは、**RequestEventArgs** 型の引数を渡します。そのプロパティは、**httpRequest** メソッドのパラメータと同じ意味と構造を持ち、要求属性を更新するように変更できます。

たとえば、認証トークンを 'Authorization'ヘッダーに入れることができます。

```
viewer.beforeSendRequest.addHandler((s, e) => {
    e.settings.requestHeaders.Authorization = 'Bearer ' + appAuthService.getToken();
});
```

ブラウザで自動的に（たとえば、URLがwindow.open（）関数へのパラメータとして、またはHTMLリンクとして使用される場合）実行されるHTTP要求が誘導するためにURLが使用されている場合、e.settings引数はnullになります。

継承元 **ViewerBase**
引数 **RequestEventArgs**

fullScreenChanged

全画面表示モードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

mouseModeChanged

マウスモードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

pageIndexChanged

ページインデックスが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

queryLoadingData

ドキュメントのロード前に、サービスに送られる要求データをクエリーしたときに発生します。

継承元 **ViewerBase**
引数 **QueryLoadingDataEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectMouseModeChanged

非推奨になりました。代わりにmouseModeChangedを使用してください。

継承元 **ViewerBase**
引数 **EventArgs**

viewModeChanged

ビューモードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

zoomFactorChanged

ズーム倍率が変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

WjReportViewer クラス

ファイル wijmo.angular2.js
モジュール wijmo/wijmo.angular2.viewer
基本クラス ReportViewer
表示 ☐継承されたメンバー ☐イベント発生元

ReportViewer コントロールに対応するAngular 2コンポーネント。

wj-report-viewerコンポーネントを使用して、Angular 2アプリケーションに**ReportViewer**コントロールを追加します。 Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjReportViewerコンポーネントは、**ReportViewer**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- beforeSendRequestNg
- controlTemplate
- filePath
- fullScreen
- fullScreenChangedNg
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- mouseMode
- mouseModeChangedNg
- pageIndex
- pageIndexChangedNg
- paginated
- parameters
- queryLoadingDataNg
- reportName
- requestHeaders
- rightToLeft
- selectMouseMode
- selectMouseModeChangedNg
- serviceUrl
- thresholdWidth
- viewMode
- viewModeChangedNg
- wjModelProperty
- zoomFactor
- zoomFactorChangedNg
- zoomMode

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getReportNames
- getReports
- getTemplate
- initialize
- invalidate
- invalidateAll
- moveToPage
- onBeforeSendRequest
- onFullScreenChanged
- onGotFocus
- onLostFocus
- onMouseModeChanged
- onPageIndexChanged
- onQueryLoadingData
- onRefreshed
- onRefreshing
- onSelectMouseModeChanged
- onViewModeChanged
- onZoomFactorChanged
- refresh
- refreshAll
- reload
- removeEventListener
- showPageSetupDialog
- zoomToView
- zoomToViewWidth

イベント

- beforeSendRequest
- fullScreenChanged
- gotFocus
- lostFocus
- mouseModeChanged
- pageIndexChanged
- queryLoadingData
- refreshed
- refreshing

- selectMouseModeChanged
- viewModeChanged
- zoomFactorChanged

コンストラクタ

constructor

```
constructor(element: any, options?: any): ReportViewer
```

ReportViewer クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: **any** OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **ReportViewer**
戻り値 **ReportViewer**

プロパティ

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

beforeSendRequestNg

プログラムによるアクセスに使用されるWijmo **beforeSendRequest**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **beforeSendRequest** Wijmoイベント名を使用してください。

型 **EventEmitter**

STATIC controlTemplate

ビューアコントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **ViewerBase**
型 **any**

filePath

サーバー上のドキュメントの完全パスを取得または設定します。

パスは、指定されたドキュメントをロードするためにサーバーに登録されているプロバイダのキーで始まります。

継承元 **ViewerBase**
型 **string**

- fullScreen

ビューアが全画面表示モードかどうかを示す値を取得または設定します。

継承元 `ViewerBase`
型 **boolean**

- fullScreenChangedNg

プログラムによるアクセスに使用されるWijmo **fullScreenChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **fullScreenChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- hostElement

コントロールをホストしているDOM要素を取得します。

継承元 `Control`
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- mouseMode

マウスの動作を示す値を取得または設定します。

デフォルトはSelectToolです。その場合は、マウスでクリックしてドラッグすると、テキストが選択されます。

継承元 `ViewerBase`
型 `MouseMode`

- mouseModeChangedNg

プログラムによるアクセスに使用されるWijmo **mouseModeChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **mouseModeChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- pageIndex

ビューパネルに現在表示されているページのインデックスを取得します。

継承元 `ViewerBase`
型 **number**

- pageIndexChangedNg

プログラムによるアクセスに使用されるWijmo **pageIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **pageIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- paginated

コンテンツを一連の固定サイズのページとして表すかどうかを示す値を取得または設定します。

デフォルト値はnullです。その場合、FlexReportではページ区切り付きモードが使用され、SSRSレポートではページ区切りなしモードが使用されます。

継承元 `ReportViewer`
型 **boolean**

- parameters

レポートの実行に使用するパラメータを記述する{name: value}ペアの辞書を取得または設定します。

このプロパティは、レポートで特定のパラメータ（たとえば、非表示のパラメータ）を初期段階で渡す必要がある場合に役に立ちます。

```
reportViewer.parameters = {
   'CustomerID': 'ALFKI'
};
```

継承元 **ReportViewer**
型 **any**

- queryLoadingDataNg

プログラムによるアクセスに使用されるWijmo **queryLoadingData**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **queryLoadingData** Wijmoイベント名を使用してください。

型 **EventEmitter**

- reportName

レポート名を取得または設定します。

FlexReportの場合は、FlexReport定義ファイルに定義されたレポート名でこれを設定します。 SSRSレポートの場合は、これを空の文字列のままにしてください。SSRSレポートのパスは、**filePath** プロパティで指定します。

継承元 **ReportViewer**
型 **string**

- requestHeaders

データを送信または要求するときに使用する要求ヘッダーを含むオブジェクトを取得または設定します。

このプロパティは、認証が必要なシナリオでよく使用されます。以下に例を示しています。

```
viewer.requestHeaders = {
    Authorization: 'Bearer ' + appAuthService.getToken();
};
```

継承元 **ViewerBase**
型 **any**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

- selectMouseMode

非推奨になりました。代わりにmouseModeを使用してください。

継承元 **ViewerBase**
型 **boolean**

- selectMouseModeChangedNg

プログラムによるアクセスに使用されるWijmo **selectMouseModeChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectMouseModeChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- serviceUrl

C1 Web APIサービスのアドレスを取得または設定します。

例："http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report"。

継承元 **ViewerBase**
型 **string**

- thresholdWidth

モバイルテンプレートとPCテンプレートの切り替えに使用するしきい値を取得または設定します。

デフォルト値は767pxです。コントロールの幅がthresholdWidthより小さい場合は、モバイルテンプレートが適用されます。コントロールの幅がthresholdWidth以上の場合は、PCテンプレートが適用されます。 thresholdWidthを0に設定すると、PCテンプレートだけが適用されます。 9999などの大きな数字に設定すると、モバイルテンプレートだけが適用されます。

継承元 **ViewerBase**
型 **number**

- viewMode

ドキュメントページの表示方法を示す値を取得または設定します。

継承元 **ViewerBase**
型 **ViewMode**

- viewModeChangedNg

プログラムによるアクセスに使用されるWijmo **viewModeChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **viewModeChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

- zoomFactor

ドキュメントページを表示する現在のズーム倍率を示す値を取得または設定します。

継承元 **ViewerBase**
型 **number**

zoomFactorChangedNg

プログラムによるアクセスに使用されるWijmo **zoomFactorChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **zoomFactorChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

zoomMode

ドキュメントページを表示する現在のズームモードを示す値を取得または設定します。

継承元 **ViewerBase**
型 **ZoomMode**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

STATIC getReportNames

```
getReportNames(serviceUrl: string, reportFilePath: string, httpHandler?: IHttpRequestHandler): IPromise
```

指定されたFlexReport定義ファイルに定義されたレポート名を取得します。

パラメーター

- **serviceUrl**: **string**
  C1 Web APIサービスのアドレス。
- **reportFilePath**: **string**
  FlexReport定義ファイルの完全パス。
- **httpHandler**: **IHttpRequestHandler** OPTIONAL
  The HTTP request handler. This parameter is optional.

継承元 **ReportViewer**
戻り値 **IPromise**

STATIC getReports

```
getReports(serviceUrl: string, path: string, data?: any, httpHandler?: IHttpRequestHandler): IPromise
```

指定されたフォルダパス内のカタログ項目を取得します。

次のデータパラメータを渡すことで、フォルダパスの下のすべての項目を取得できます。 1) true値。 2) true値を含む"recursive"プロパティを持つオブジェクト。

パラメーター

- **serviceUrl**: **string**
  C1 Web APIサービスのアドレス。
- **path**: **string**
  フォルダパス。 FlexReport定義ファイルのパスは、フォルダパスとして扱われます。
- **data**: **any** OPTIONAL
  レポートサービスに送られる要求データ、またはパス下のすべての項目を取得するかどうかを示すboolean値。
- **httpHandler**: **IHttpRequestHandler** OPTIONAL
  The HTTP request handler. This parameter is optional.

継承元 `ReportViewer`
戻り値 `IPromise`

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 `Control`
戻り値 `string`

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

moveToPage

```
moveToPage(index: number): IPromise
```

指定したインデックスにあるページに移動します。

パラメーター

- **index**: **number**
  移動先のページの0から始まるインデックス。

継承元 **ViewerBase**
戻り値 **IPromise**

onBeforeSendRequest

```
onBeforeSendRequest(e: RequestEventArgs): void
```

beforeSendRequest イベントを発生させます。

パラメーター

- **e**: **RequestEventArgs**
  RequestEventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onFullScreenChanged

```
onFullScreenChanged(e?: EventArgs): void
```

fullScreenChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onMouseModeChanged

```
onMouseModeChanged(e?: EventArgs): void
```

mouseModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onPageIndexChanged

```
onPageIndexChanged(e?: EventArgs): void
```

pageIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onQueryLoadingData

```
onQueryLoadingData(e: QueryLoadingDataEventArgs): void
```

queryLoadingData イベントを発生させます。

パラメーター

- **e**: **QueryLoadingDataEventArgs**
  ロード中のデータを含む**QueryLoadingDataEventArgs** オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectMouseModeChanged

```
onSelectMouseModeChanged(e?: EventArgs): void
```

非推奨になりました。代わりにonMouseModeChangedを使用してください。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onViewModeChanged

```
onViewModeChanged(e?: EventArgs): void
```

viewModeChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

onZoomFactorChanged

```
onZoomFactorChanged(e?: EventArgs): void
```

zoomFactorChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL
  EventArgs オブジェクト。

継承元 **ViewerBase**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **ViewerBase**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

reload

```
reload(): void
```

ドキュメントを再ロードします。

これは、ドキュメントを強制的に再ロードおよび再レンダリングする場合に便利です。

継承元 **ViewerBase**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

showPageSetupDialog

```
showPageSetupDialog(): void
```

ページ設定ダイアログボックスを表示します。

継承元 **ViewerBase**
戻り値 **void**

zoomToView

```
zoomToView(): void
```

現在のページを拡大縮小して、ビューパネルにページ全体を表示します。

継承元 **ViewerBase**
戻り値 **void**

zoomToViewWidth

```
zoomToViewWidth(): void
```

ビューパネルの幅に合わせて現在のページを拡大縮小します。

継承元 **ViewerBase**
戻り値 **void**

イベント

beforeSendRequest

各要求がサーバーに送信される前に発生します。

このイベントでは、サーバーに送信する前に、URL、ヘッダー、データなどのリクエストオプションに加えてリクエストメソッドも変更できます。 このイベントは、**RequestEventArgs** 型の引数を渡します。そのプロパティは、**httpRequest** メソッドのパラメータと同じ意味と構造を持ち、要求属性を更新するように変更できます。

たとえば、認証トークンを 'Authorization'ヘッダーに入れることができます。

```
viewer.beforeSendRequest.addHandler((s, e) => {
    e.settings.requestHeaders.Authorization = 'Bearer ' + appAuthService.getToken();
});
```

ブラウザで自動的に（たとえば、URLがwindow.open（）関数へのパラメータとして、またはHTMLリンクとして使用される場合）実行されるHTTP要求が誘導するためにURLが使用されている場合、e.settings引数はnullになります。

継承元 **ViewerBase**
引数 **RequestEventArgs**

fullScreenChanged

全画面表示モードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

mouseModeChanged

マウスモードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

pageIndexChanged

ページインデックスが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

queryLoadingData

ドキュメントのロード前に、サービスに送られる要求データをクエリーしたときに発生します。

継承元 **ViewerBase**
引数 **QueryLoadingDataEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectMouseModeChanged

非推奨になりました。代わりにmouseModeChangedを使用してください。

継承元 **ViewerBase**
引数 **EventArgs**

viewModeChanged

ビューモードが変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

zoomFactorChanged

ズーム倍率が変更された後に発生します。

継承元 **ViewerBase**
引数 **EventArgs**

wijmo/wijmo.angular2.nav モジュール

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.nav**

wijmo.navモジュールに対応するAngular 2コンポーネントを含みます。

wijmo.angular2.navは、アンビエントモジュール名を使用してコードにインポートできる 外部TypeScriptモジュールです。次に例を示します。

```
import * as wjNav from 'wijmo/wijmo.angular2.nav';

@Component({
    directives: [wjNav.WjTreeView],
    template: `
      <wj-tree-view [itemsSource]="items" [displayMemberPath]="'header'" [childItemsPath]="'items'">
      </wj-tree-view;`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

クラス

- WjTab
- WjTabPanel
- WjTreeView

WjTab クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.nav**
基本クラス **Tab**
表示 ☐継承されたメンバー ☐イベント発生元

Tab コントロールに対応するAngular 2コンポーネント。

wj-tabコンポーネントは、 **WjTabPanel** コンポーネントに含める必要があります。

wj-tabコンポーネントを使用して、Angular 2アプリケーションに**Tab**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjTabコンポーネントは、**Tab**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- header
- initialized
- isDisabled
- isInitialized
- isVisible
- pane
- tabPanel
- wjProperty

メソッド

- created

コンストラクタ

constructor

```
constructor(header: any, pane: any): Tab
```

Tab クラスの新しいインスタンスを初期化します。

パラメーター

- **header**: **any**
  タブヘッダーを含む要素または、要素に該当するCSSセレクター。

- **pane**: **any**
  タブの内容を含む要素または要素に該当するCSSセレクター。

継承元 **Tab**
戻り値 **Tab**

プロパティ

- header

タブのヘッダ要素を取得します。

継承元 **Tab**
型 **HTMLElement**

- initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

- isDisabled

この **Tab** が無効かどうかを示す値を取得または設定します。

継承元 **Tab**
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、**initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isVisible

この**Tab** が表示されるかどうかを示す値を取得または設定します。

継承元 **Tab**
型 **boolean**

pane

タブの内容要素を取得します。

継承元 **Tab**
型 **HTMLElement**

- tabPanel

このタブを含む**`TabPanel`** への参照を取得します。

継承元 **Tab**
型 **`TabPanel`**

- wjProperty

このコンポーネントの割り当て先のプロパティの名前を取得または設定します。 デフォルト値は'tabs'です。

型 **string**

メソッド

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

WjTabPanel クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.nav**
基本クラス **TabPanel**
表示 ☐継承されたメンバー ☐イベント発生元

TabPanel コントロールに対応するAngular 2コンポーネント。

wj-tab-panelコンポーネントを使用して、Angular 2アプリケーションに**TabPanel**コントロールを 追加できます。Angular 2マークアップの構文については、 「Angular 2マークアップ構文」を参照してください。

WjTabPanelコンポーネントは、**TabPanel**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。 **wj-tab-panel**コンポーネントには、**WjTab** 子コンポーネントを含めることができます。

コンストラクタ

- constructor

プロパティ

- asyncBindings
- autoSwitch
- controlTemplate
- gotFocusNg
- hostElement
- initialized
- isAnimated
- isDisabled
- isInitialized
- isTouching
- isUpdating
- lostFocusNg
- rightToLeft
- selectedIndex
- selectedIndexChangedNg
- selectedTab
- tabs
- wjModelProperty

メソッド

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTab
- getTemplate

- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRefreshed
- onRefreshing
- onSelectedIndexChanged
- refresh
- refreshAll
- removeEventListener

イベント

- gotFocus
- lostFocus
- refreshed
- refreshing
- selectedIndexChanged

コンストラクタ

constructor

```
constructor(element: any, options?, keepChildren?: boolean): TabPanel
```

TabPanel クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。
- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。
- **keepChildren**: **boolean** OPTIONAL
  子要素を保持するかどうか。 trueに設定した場合、呼び出し元はDOMに基づいて**tabs** の配列に値を設定します。

継承元 **TabPanel**
戻り値 **TabPanel**

プロパティ

- asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoSwitch

ユーザーが矢印キーを使用してタブを選択したときにコントロールがタブを自動的に切り替えるかどうかを決定する値を取得または設定します。

autoSwitch がtrue（デフォルト値）に設定されている場合、矢印キーを押すとタブが自動的に切り替わります。タブキーを押すと、選択されていないタブヘッダーは除外され、タブ順序の次の要素が選択されます。

autoSwitch がfalseに設定されている場合、矢印キーまたはTabキーを押すと、フォーカスが次のタブヘッダーまたは前のタブヘッダーに移動しますが、タブは切り替わりません。フォーカスがあるタブをアクティブにするには、EnterキーまたはSpaceキーを押す必要があります。

ほとんどの場合、デフォルト値は適切な（アクセス可能な）動作を提供しますが、 **autoSwitch** をfalseに設定したいユーザーも存在するかもしれません。 このトピックの詳細については、以下をご参考ください。 **W3C ARIA practices** と **SimplyAccessible** articles.

継承元 **TabPanel**
型 **boolean**

STATIC controlTemplate

TabPanel コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **TabPanel**
型 **any**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **Control**
型 **HTMLElement**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isAnimated

タブの変更をフェードイン効果でアニメーション化するかどうかを決定する値を取得または設定します。

The default value for this property is **true**.

継承元 **TabPanel**
型 **boolean**

- isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 `Control`
型 **boolean**

- isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

- isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

- rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 `Control`
型 **boolean**

- selectedIndex

現在選択されている（アクティブな）タブのインデックスを取得または設定します。

継承元 `TabPanel`
型 **number**

- selectedIndexChangedNg

プログラムによるアクセスに使用されるWijmo **selectedIndexChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedIndexChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

- selectedTab

現在選択されている**Tab** を取得または設定します。

継承元 **TabPanel**
型 **Tab**

- tabs

header と**pane** プロパティが**TabPanel** コントロールの内容を決定する**Tab** オブジェクトの配列を取得します。

継承元 **TabPanel**
型 **ObservableArray**

- wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。
- **type**: **string**
  イベントを指定する文字列。
- **fn**: **any**
  イベントが発生したときに実行する関数。
- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.
- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getTab

```
getTab(id: string): Tab
```

IDまたはヘッダの内容によって**Tab** を取得します。

パラメーター

- **id**: **string**
  取得する**TabのID**。

継承元 **TabPanel**
戻り値 **Tab**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 `Control`
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 `Control`
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

selectedIndexChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **TabPanel**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

コントロールを更新します。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

イベント

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元 **Control**
引数 **EventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedIndexChanged

selectedIndex プロパティの値が変更されたときに発生します。

継承元 **TabPanel**
引数 **EventArgs**

WjTreeView クラス

ファイル wijmo.angular2.js
モジュール **wijmo/wijmo.angular2.nav**
基本クラス **TreeView**
表示 ☐継承されたメンバー ☐イベント発生元

TreeView コントロールに対応するAngular 2コンポーネント。

wj-tree-viewコンポーネントを使用して、Angular 2アプリケーションに**TreeView**コントロールを追加できます。 Angular 2マークアップの構文については、「Angular 2マークアップ」を参照してください。

WjTreeViewコンポーネントは、**TreeView**コントロールから派生され、 そのすべてのプロパティ、イベント、およびメソッドを継承しています。

コンストラクタ

- constructor

プロパティ

- allowDragging
- asyncBindings
- autoCollapse
- checkedItems
- checkedItemsChangedNg
- childItemsPath
- controlTemplate
- displayMemberPath
- dragEndNg
- dragOverNg
- dragStartNg
- dropNg
- expandOnClick
- formatItemNg
- gotFocusNg
- hostElement
- imageMemberPath
- initialized
- isAnimated
- isCheckedChangedNg
- isCheckedChangingNg
- isCollapsedChangedNg
- isCollapsedChangingNg
- isContentHtml
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- itemClickedNg
- itemsSource
- itemsSourceChangedNg
- lazyLoadFunction
- loadedItemsNg

- loadingItemsNg
- lostFocusNg
- nodeEditEndedNg
- nodeEditEndingNg
- nodeEditStartedNg
- nodeEditStartingNg
- nodes
- rightToLeft
- selectedItem
- selectedItemChangedNg
- selectedNode
- selectedPath
- showCheckboxes
- totalItemCount
- wjModelProperty

メソッド

- addChildNode
- addEventListener
- applyTemplate
- beginUpdate
- checkAllItems
- collapseToLevel
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getControl
- getFirstNode
- getLastNode
- getNode
- getTemplate
- initialize
- invalidate
- invalidateAll
- loadTree
- onCheckedItemsChanged
- onDragEnd
- onDragOver
- onDragStart
- onDrop
- onFormatItem
- onGotFocus
- onIsCheckedChanged

- onIsCheckedChanging
- onIsCollapsedChanged
- onIsCollapsedChanging
- onItemClicked
- onItemsSourceChanged
- onLoadedItems
- onLoadingItems
- onLostFocus
- onNodeEditEnded
- onNodeEditEnding
- onNodeEditStarted
- onNodeEditStarting
- onRefreshed
- onRefreshing
- onSelectedItemChanged
- refresh
- refreshAll
- removeEventListener
- startEditing

イベント

- checkedItemsChanged
- dragEnd
- dragOver
- dragStart
- drop
- formatItem
- gotFocus
- isCheckedChanged
- isCheckedChanging
- isCollapsedChanged
- isCollapsedChanging
- itemClicked
- itemsSourceChanged
- loadedItems
- loadingItems
- lostFocus
- nodeEditEnded
- nodeEditEnding
- nodeEditStarted
- nodeEditStarting
- refreshed
- refreshing
- selectedItemChanged

コンストラクタ

constructor

```
constructor(element: any, options?): TreeView
```

TreeView クラスの新しいインスタンスを初期化します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

- **options**: OPTIONAL
  コントロールの初期化データを含むJavaScriptオブジェクト。

継承元 **TreeView**
戻り値 **TreeView**

プロパティ

allowDragging

ユーザーが**TreeView** 内でノードをドラッグアンドドロップできるかどうかを指定する値を取得または設定します。

継承元 **TreeView**
型 **boolean**

asyncBindings

この特定のコンポーネントのグローバルな**WjOptions.asyncBindings**の設定をオーバーライドできます。 詳細については、**WjOptions.asyncBindings** プロパティの説明を参照してください。

型 **boolean**

autoCollapse

ノードが展開されたときに、兄弟ノードを折りたたむかどうかを指定する値を 取得または設定します。

このプロパティは、デフォルトではtrueに設定されています。 通常は、使用していないノードを折りたたんだ方がUIがすっきりするためです。

継承元 **TreeView**
型 **boolean**

checkedItems

現在オンに設定されている項目が含まれる配列を取得します。

返される配列には、子を持たない項目だけが含まれます。 これは、親項目のチェックボックスが子項目のオンまたはオフに 使用されるからです。

showCheckboxes プロパティと **checkedItemsChanged** プロパティも参照してください。

次に例を示します。

```
var treeViewChk = new wijmo.input.TreeView('#gsTreeViewChk', {
   displayMemberPath: 'header',
   childItemsPath: 'items',
   showCheckboxes: true,
   itemsSource: items,
   checkedItemsChanged: function (s, e) {
       var items = s.checkedItems,
           msg = '';
       if (items.length) {
           msg = '<p><b>Selected Items:</b></p><ol>\r\n';
           for (var i = 0; i < items.length; i++) {
               msg += '<li>' + items[i].header + '</li>\r\n';
           }
           msg += '</ol>';
       }
       document.getElementById('gsTreeViewChkStatus').innerHTML = msg;
   }
});
```

継承元 **TreeView**
型 **any**[]

checkedItemsChangedNg

プログラムによるアクセスに使用されるWijmo **checkedItemsChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **checkedItemsChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

childItemsPath

各ノードの子項目を含む1つ以上のプロパティの名前を 取得または設定します。

このプロパティのデフォルト値は文字列'items'です。

ほとんどの場合、子項目を含むプロパティは、 ツリーのすべてのデータ項目で同じです。このような場合は、 **childItemsPath** にその名前を設定します。

ただし、項目のレベルごとに異なるプロパティを使用して 子項目を保存する場合もあります。たとえば、カテゴリ、製品、注文から成る ツリーがあるとします。その場合に、 **childItemsPath** に次のような配列を設定します。

```
// カテゴリには製品が含まれ、製品には注文が含まれます。
tree.childItemsPath = [ 'Products', 'Orders' ];
```

継承元 **TreeView**
型 **any**

STATIC controlTemplate

TreeView コントロールのインスタンス化に使用されるテンプレートを取得または設定します。

継承元 **TreeView**
型 **any**

displayMemberPath

ノードのビジュアル表現として使用される1つ以上のプロパティの名前を 取得または設定します。

このプロパティのデフォルト値は文字列'header'です。

ほとんどの場合、ノードテキストを含むプロパティは、 ツリーのすべてのデータ項目で同じです。このような場合は、 **displayMemberPath** にその名前を設定します。

ただし、項目のレベルごとに異なるプロパティを使用して ノードテキストを表す場合もあります。たとえば、カテゴリ、製品、注文から成る ツリーがあるとします。その場合に、 **displayMemberPath** に次のような配列を設定します。

```
// カテゴリ、製品、注文のヘッダーがそれぞれ異なります。
tree.displayMemberPath = [ 'CategoryName', 'ProductName', 'OrderID' ];
```

継承元 **TreeView**
型 **any**

dragEndNg

プログラムによるアクセスに使用されるWijmo **dragEnd**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **dragEnd** Wijmoイベント名を使用してください。

型 **EventEmitter**

dragOverNg

プログラムによるアクセスに使用されるWijmo **dragOver**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **dragOver** Wijmoイベント名を使用してください。

型 **EventEmitter**

dragStartNg

プログラムによるアクセスに使用されるWijmo **dragStart**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **dragStart** Wijmoイベント名を使用してください。

型 **EventEmitter**

dropNg

プログラムによるアクセスに使用されるWijmo **drop**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **drop** Wijmoイベント名を使用してください。

型 **EventEmitter**

expandOnClick

ユーザーがノードヘッダーをクリックしたときに、折りたたまれているノードを展開するかどうかを指定する値を取得または設定します。

継承元 **TreeView**
型 **boolean**

formatItemNg

プログラムによるアクセスに使用されるWijmo **formatItem**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **formatItem** Wijmoイベント名を使用してください。

型 **EventEmitter**

gotFocusNg

プログラムによるアクセスに使用されるWijmo **gotFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **gotFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

hostElement

コントロールをホストしているDOM要素を取得します。

継承元 **`Control`**
型 **HTMLElement**

imageMemberPath

ノードの画像のソースとして使用する1つ以上のプロパティの名前を 取得または設定します。

継承元 **`TreeView`**
型 **any**

initialized

このイベントは、コンポーネントがAngularによって初期化された後、つまり すべての連結プロパティが割り当てられ、子コンポーネント（ある場合）が初期化された後にトリガされます。

型 **EventEmitter**

isAnimated

ノードを展開または折りたたむときにアニメーションを使用するかどうかを示す値を取得または設定します。

The default value for this property is **true**.

継承元 **`TreeView`**
型 **boolean**

isCheckedChangedNg

プログラムによるアクセスに使用されるWijmo **isCheckedChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isCheckedChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

isCheckedChangingNg

プログラムによるアクセスに使用されるWijmo **isCheckedChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isCheckedChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

isCollapsedChangedNg

プログラムによるアクセスに使用されるWijmo **isCollapsedChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isCollapsedChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

isCollapsedChangingNg

プログラムによるアクセスに使用されるWijmo **isCollapsedChanging**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **isCollapsedChanging** Wijmoイベント名を使用してください。

型 **EventEmitter**

isContentHtml

項目をプレーンテキストまたはHTMLに連結するかどうかを示す値を取得または設定します。

The default value for this property is **false**.

継承元 **TreeView**
型 **boolean**

isDisabled

コントロールが無効かどうかを判定する値を取得または設定します。

無効化されたコントロールは、マウスイベントやキーボードイベントを取得できません。

継承元 **Control**
型 **boolean**

isInitialized

コンポーネントがAngularによって初期化されているかどうかを示します。 この値は、 **initialized**イベントをトリガする直前にfalseからtrueになります。

型 **boolean**

isReadOnly

ユーザーがノードのテキストを編集できるかどうかを指定する値を取得または設定します。

isReadOnly プロパティをfalseに設定すると、ノードに直接入力することで ツリーノードのコンテンツを編集することができます。ノードのコンテンツ全体を選択した状態で、 ［F2］キーを使用して編集モードに入ることもできます。

次のメソッドとイベントを使用して、編集動作をカスタマイズできます。

メソッド：**startEditing**、**finishEditing**

イベント：**nodeEditStarting**、**nodeEditStarted**、**nodeEditEnding**、 **nodeEditEnded**

The default value for this property is **true**.

継承元 **TreeView**
型 **boolean**

isTouching

現在、コントロールがタッチイベントを処理しているかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

isUpdating

コントロールが現在更新中かどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

itemClickedNg

プログラムによるアクセスに使用されるWijmo **itemClicked**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemClicked** Wijmoイベント名を使用してください。

型 **EventEmitter**

itemsSource

TreeView 項目を含む配列を取得または設定します。

TreeView #see:itemsSource 配列は一般に、 子項目を含む項目から成る階層構造を持ちます。項目の深さに制限はありません。

たとえば、次の配列は、それぞれが2つの子ノードを持つ 3つの最上位ノードを含むツリーを生成します。

```
var tree = new wijmo.input.TreeView('#treeView', {
    displayMemberPath: 'header',
    childItemsPath: 'items',
    itemsSource: [
        { header: '1 first', items: [
            { header: '1.1 first child' },
            { header: '1.2 second child' },
        ] },
        { header: '2 second', items: [
            { header: '3.1 first child' },
            { header: '3.2 second child' },
        ] },
        { header: '3 third', items: [
            { header: '3.1 first child' },
            { header: '3.2 second child' },
        ] }
    ]
});
```

継承元 **TreeView**
型 **any**[]

itemsSourceChangedNg

プログラムによるアクセスに使用されるWijmo **itemsSourceChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **itemsSourceChanged** Wijmo イベント名を使用してください。

型 **EventEmitter**

lazyLoadFunction

オンデマンドで子ノードをロードする関数を取得または設定します。

lazyLoadFunction は、2つのパラメータとして、展開されるノードと、 データが取得可能になったときに呼び出されるコールバックを 受け取ります。

コールバック関数は、**TreeView** にノードのロードプロセスが 完了したことを通知します。データのロード時にエラーがあっても、 コールバックは必ず呼び出されます。

次に例を示します。

```
var treeViewLazyLoad = new wijmo.input.TreeView('#treeViewLazyLoad', {
   displayMemberPath: 'header',
   childItemsPath: 'items',
   itemsSource: [ // 3つの遅延ロードノードから開始します
       { header: 'Lazy Node 1', items: []},
       { header: 'Lazy Node 2', items: [] },
       { header: 'Lazy Node 3', items: [] }
   ],
   lazyLoadFunction: function (node, callback) {
       setTimeout(function () { // httpの遅延をシミュレーションします
           var result = [ // 結果をシミュレーションします
               { header: 'Another lazy node...', items: [] },
               { header: 'A non-lazy node without children' },
               { header: 'A non-lazy node with child nodes', items: [
                 { header: 'hello' },
                 { header: 'world' }
               ]}
           ];
           callback(result); // コントロールに結果を返します
       }, 2500); // 2.5秒のhttpの遅延をシミュレーションします
   }
});
```

遅延ロードノードを含むツリーには、それぞれのノードにチェックボックスが ないことがある（**showCheckboxes** プロパティを参照）、**collpaseToLevel** メソッドがまだロードされていない折りたたみ ノードを展開しない、といった制限があります。

継承元 **TreeView**
型 **Function**

loadedItemsNg

プログラムによるアクセスに使用されるWijmo **loadedItems**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadedItems** Wijmoイベント名を使用してください。

型 **EventEmitter**

loadingItemsNg

プログラムによるアクセスに使用されるWijmo **loadingItems**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **loadingItems** Wijmoイベント名を使用してください。

型 **EventEmitter**

lostFocusNg

プログラムによるアクセスに使用されるWijmo **lostFocus**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **lostFocus** Wijmoイベント名を使用してください。

型 **EventEmitter**

● nodeEditEndedNg

プログラムによるアクセスに使用されるWijmo **nodeEditEnded**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **nodeEditEnded** Wijmoイベント名を使用してください。

型 **EventEmitter**

● nodeEditEndingNg

プログラムによるアクセスに使用されるWijmo **nodeEditEnding**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **nodeEditEnding** Wijmoイベント名を使用してください。

型 **EventEmitter**

● nodeEditStartedNg

プログラムによるアクセスに使用されるWijmo **nodeEditStarted**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **nodeEditStarted** Wijmoイベント名を使用してください。

型 **EventEmitter**

● nodeEditStartingNg

プログラムによるアクセスに使用されるWijmo **nodeEditStarting**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **nodeEditStarting** Wijmoイベント名を使用してください。

型 **EventEmitter**

● nodes

現在ロードされているノードを表す**TreeNode** オブジェクトの配列を取得します。

継承元 **TreeView**
型 **TreeNode[]**

● rightToLeft

要素内のコントロールを右から左のレイアウトでホストするかどうかを示す値を取得します。

継承元 **Control**
型 **boolean**

● selectedItem

現在選択されているデータ項目を取得または設定します。

継承元 **TreeView**
型 **any**

selectedItemChangedNg

プログラムによるアクセスに使用されるWijmo **selectedItemChanged**イベントのAngular（EventEmitter）バージョン。 コードでこのイベントのAngularバージョンをサブスクライブする場合は、このイベント名を使用してください。 テンプレート連結では、通常の **selectedItemChanged** Wijmoイベント名を使用してください。

型 **EventEmitter**

selectedNode

現在選択されている**TreeNode** を取得または設定します。

継承元 **TreeView**
型 **TreeNode**

selectedPath

ルートから現在選択されているノードまでのすべてのノードのテキストが 入った配列を取得します。

継承元 **TreeView**
型 **string**[]

showCheckboxes

TreeView で、ノードにチェックボックスを追加し、その状態を管理するかどうかを決定する値を 取得または設定します。

このプロパティは、遅延ロードノードがないツリーでのみ使用できます （**lazyLoadFunction** プロパティを参照）。

checkedItems プロパティと**checkedItemsChanged** イベントも参照してください。

The default value for this property is **false**.

継承元 **TreeView**
型 **boolean**

totalItemCount

ツリー内の項目の総数を取得します。

継承元 **TreeView**
型 **number**

wjModelProperty

[(ngModel)]ディレクティブ（指定されている場合）によって表されるプロパティの名前を定義します。 デフォルト値は"です。

型 **string**

メソッド

addChildNode

```
addChildNode(index: number, dataItem: any): TreeNode
```

特定の位置に子ノードを追加します。

パラメーター

- **index**: **number**
  新しい子ノードのインデックス。

- **dataItem**: **any**
  新しいノードの作成に使用されたデータ項目。

継承元 **TreeView**
戻り値 **TreeNode**

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean, passive?: boolean): void
```

この**Control** が所有する要素にイベントリスナーを追加します。

コントロールは、アタッチされているリスナーとそのハンドラのリストを保持し、コントロール が破棄されているときにそれらを簡単に削除すること ができます（**dispose** と **removeEventListener** メソッドを参照してください）。

イベントリスナーを削除しないと、メモリリークが発生する可能性があります。

The **passive** parameter is set to false by default, which means the event handler may call **event.preventDefault()**. If you are adding passive handlers to touch or wheel events, setting this parameter to true will improve application performance.

For details on passive event listeners, please see **Improving scrolling performance with passive listeners**.

パラメーター

- **target**: **EventTarget**
  イベントのターゲット要素。

- **type**: **string**
  イベントを指定する文字列。

- **fn**: **any**
  イベントが発生したときに実行する関数。

- **capture**: **boolean** OPTIONAL
  Whether the listener should be handled by the control before it is handled by the target element.

- **passive**: **boolean** OPTIONAL
  Indicates that the handler will never call **preventDefault()**.

継承元 **Control**
戻り値 **void**

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

コントロールの新しいインスタンスにテンプレートを適用し、ルート要素を返します。

このメソッドはテンプレート化されたコントロールのコンストラクターによって呼び出され、 テンプレートのパーツを対応するコントロールメンバにバインドする役割を持ちます。

以下のサンプルコードは、**InputNumber** コントロールのインスタンスにテンプレートを適用します。このテンプレートには、'wj-part'属性が'input'、'btn-inc'、および'btn-dec'に設定された要素を含める必要があります。 コントロールのメンバである'_tbx'、'_btnUp'、'_btnDn'には、これらの要素への参照が割り当てられます。

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

パラメーター

- **classNames**: **string**
  コントロールのホスト要素に追加するクラスの名前。
- **template**: **string**
  コントロールのテンプレートを定義するHTML文字列。
- **parts**: **Object**
  パーツ変数とその名前のディクショナリー。
- **namePart**: **string** OPTIONAL
  ホスト要素にちなんで名付けられたパーツの名前。 これにより、コントロールがフォームで使用されたときにデータをどのように送信するかが決まります。

継承元 **Control**
戻り値 **HTMLElement**

beginUpdate

```
beginUpdate(): void
```

次に**endUpdate** が呼び出されるまで通知を中断します。

継承元 **Control**
戻り値 **void**

checkAllItems

```
checkAllItems(check: boolean): void
```

ツリーのすべてのチェックボックスをオンまたはオフにします。

パラメーター

- **check**: **boolean**
  すべてのチェックボックスをオンまたはオフにするかどうか。

継承元 **TreeView**
戻り値 **void**

collapseToLevel

```
collapseToLevel(level: number): void
```

すべてのツリー項目を指定されたレベルまで折りたたみます。

通常、このメソッドは一度に複数のノードを展開または折りたたみます。ただし、どのノードにも遅延ロードを実行しないので、 遅延ロードする必要がある折りたたみノードは展開されません。

パラメーター

- **level**: **number**
  表示する最大ノードレベル。

継承元 **TreeView**
戻り値 **void**

containsFocus

```
containsFocus(): boolean
```

このコントロールにフォーカスのある要素が含まれているかどうかをチェックします。

継承元 **Control**
戻り値 **boolean**

created

```
created(): void
```

Wijmoコンポーネントから継承されるカスタムコンポーネントを作成する場合は、このメソッドを オーバーライドして、通常はクラスコンストラクタで行う必要な初期化を行うことができます。 このメソッドは、Wijmoコンポーネントコンストラクタの最後の行で呼び出され、 カスタムコンポーネントのコンストラクタを宣言しなくて済むようにします。これにより、 コンストラクタのパラメータを保持したり、Wijmoコンポーネントのコンストラクタパラメータとの同期を維持する必要がなくなります。

戻り値 **void**

deferUpdate

```
deferUpdate(fn: Function): void
```

beginUpdate/endUpdateブロック内で関数を実行します。

この関数の実行が完了するまでコントロールは更新されません。 このメソッドは、関数が例外を生成した場合でも**endUpdate** が呼び出されるようにします。

パラメーター

- **fn**: **Function**
  実行する関数。

継承元 **Control**
戻り値 **void**

dispose

```
dispose(): void
```

ホスト要素との関連付けを解除することによってコントロールを破棄します。

dispose メソッドは、**addEventListener** メソッドによって追加されたイベントリスナーを自動的に削除します。

コントロールを動的に作成および削除するアプリケーションでは、**dispose** メソッドを呼び出すことが重要です。コントロールを破棄しないと、メモリリークが発生する可能があります。

継承元 **Control**
戻り値 **void**

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

HTML要素に含まれるすべてのWijmoコントロールを破棄します。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。

継承元 **Control**
戻り値 **void**

endUpdate

```
endUpdate(): void
```

beginUpdate の呼び出しによって中断された通知を再開します。

継承元 **Control**
戻り値 **void**

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

編集中の値を確定して編集モードを終了します。

パラメーター

- **cancel**: **boolean** OPTIONAL
  保留中の編集をキャンセルするかコミットするか。

継承元 **TreeView**
戻り値 **boolean**

focus

```
focus(): void
```

このコントロールにフォーカスを設定します。

継承元 **Control**
戻り値 **void**

STATIC getControl

```
getControl(element: any): Control
```

指定したDOM要素でホストされているコントロールを取得します。

パラメーター

- **element**: **any**
  コントロールをホストするDOM要素、またはホスト要素のCSS セレクター（例: '#theCtrl'）。

継承元 **Control**
戻り値 **Control**

getFirstNode

```
getFirstNode(visible?: boolean, enabled?: boolean): TreeNode
```

TreeView の最初の**TreeNode** への参照を取得します。

パラメーター

- **visible**: **boolean** OPTIONAL
  可視のノード（祖先ノードが折りたたまれていない）だけを返すかどうか。

- **enabled**: **boolean** OPTIONAL
  有効なノード（祖先ノードが無効でない）だけを返すかどうか。

継承元 **TreeView**
戻り値 **TreeNode**

getLastNode

```
getLastNode(visible?: boolean, enabled?: boolean): TreeNode
```

TreeView の最後の**TreeNode** への参照を取得します。

パラメーター

- **visible**: **boolean** OPTIONAL
  可視のノード（祖先ノードが折りたたまれていない）だけを返すかどうか。

- **enabled**: **boolean** OPTIONAL
  有効なノード（祖先ノードが無効でない）だけを返すかどうか。

継承元 **TreeView**
戻り値 **TreeNode**

getNode

```
getNode(item: any): TreeNode
```

特定のデータ項目を表す**TreeNode** オブジェクトを取得します。

パラメーター

- **item**: **any**
  検索するデータ項目。

継承元 **TreeView**
戻り値 **TreeNode**

getTemplate

```
getTemplate(): string
```

コントロールのインスタンスの作成に使用されたHTMLテンプレートを取得します。

このメソッドは、クラス階層をさかのぼってコントロールのテンプレートを指定する最も近い祖先を探します。たとえば、 **ComboBox** コントロールのプロトタイプを指定した場合、 そのプロトタイプによって**DropDown** 基本クラスで定義されたテンプレートがオーバーライドされます。

継承元 **Control**
戻り値 **string**

initialize

```
initialize(options: any): void
```

指定したオブジェクトからプロパティをコピーしてコントロールを初期化します。

このメソッドを使用すると、各プロパティの値をコードで設定する代わりにプレーンなデータオブジェクトを使用してコントロールを初期化できます。

例:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// これは以下と同等です。
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// など
```

初期化データは適用時に型チェックされます。初期化オブジェクトに不明なプロパティ名または無効なデータ型が含まれている場合、このメソッドは例外をスローします。

パラメーター

- **options**: **any**
  初期化データを含むオブジェクト。

継承元 **Control**
戻り値 **void**

invalidate

```
invalidate(fullUpdate?: boolean): void
```

非同期更新を発生させるため、コントロールを無効にします。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうか。

継承元 **Control**
戻り値 **void**

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

指定したHTML要素に含まれるすべてのWijmoコントロールを無効化します。

このメソッドは、コントロールの表示状態やサイズを変更する動的なパネルをアプリケーションで使用している場合に使用します。たとえば、スプリッタ、アコーディオン、およびタブコントロールは通常、その中の要素の表示状態を変更します。 この場合、その要素に含まれるコントロールに通知しないと、それらのコントロールが適切に機能しなくなる可能性があります。

これが起こる場合は、動的コンテナーで適切なイベントを処理し、 **invalidateAll** メソッドを呼び出してコンテナー内のWijmoコントロールのレイアウト情報が適切に更新されるようにする必要があります。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

loadTree

```
loadTree(preserveOutlineState?: boolean): void
```

現在の**itemsSource** のデータを使用してツリーをロードします。

パラメーター

- **preserveOutlineState**: **boolean** OPTIONAL
  ツリーのデータをロードするときにアウトラインの状態を保持するかどうか。 デフォルト値はfalseです。

継承元 **TreeView**
戻り値 **void**

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

checkedItemsChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **TreeView**
戻り値 **void**

onDragEnd

```
onDragEnd(e?: EventArgs): void
```

dragEnd イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **TreeView**
戻り値 **void**

onDragOver

```
onDragOver(e: TreeNodeDragDropEventArgs): boolean
```

dragOver イベントを発生させます。

パラメーター

- **e**: **TreeNodeDragDropEventArgs**
  イベントデータを含む**TreeNodeDragDropEventArgs**。

継承元 **TreeView**
戻り値 **boolean**

onDragStart

```
onDragStart(e: TreeNodeEventArgs): boolean
```

dragStart イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

継承元 **TreeView**
戻り値 **boolean**

onDrop

```
onDrop(e: TreeNodeDragDropEventArgs): boolean
```

drop イベントを発生させます。

パラメーター

- **e**: **TreeNodeDragDropEventArgs**
  イベントデータを含む**TreeNodeDragDropEventArgs**。

継承元 **TreeView**
戻り値 **boolean**

onFormatItem

```
onFormatItem(e: FormatNodeEventArgs): void
```

formatItem イベントを発生させます。

パラメーター

- **e**: **FormatNodeEventArgs**
  イベントデータを含む**FormatNodeEventArgs**。

継承元 **TreeView**
戻り値 **void**

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

gotFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onIsCheckedChanged

```
onIsCheckedChanged(e: TreeNodeEventArgs): void
```

isCheckedChanged イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

継承元 **TreeView**
戻り値 **void**

onIsCheckedChanging

```
onIsCheckedChanging(e: TreeNodeEventArgs): boolean
```

isCheckedChanging イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

継承元 **TreeView**
戻り値 **boolean**

onIsCollapsedChanged

```
onIsCollapsedChanged(e: TreeNodeEventArgs): void
```

isCollapsedChanged イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

継承元 **TreeView**
戻り値 **void**

onIsCollapsedChanging

```
onIsCollapsedChanging(e: TreeNodeEventArgs): boolean
```

isCollapsedChanging イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

継承元 **TreeView**
戻り値 **boolean**

onItemClicked

```
onItemClicked(e?: EventArgs): void
```

itemClicked イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **TreeView**
戻り値 **void**

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

itemsSourceChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **TreeView**
戻り値 **void**

onLoadedItems

```
onLoadedItems(e?: EventArgs): void
```

loadedItems イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **TreeView**
戻り値 **void**

onLoadingItems

```
onLoadingItems(e?: CancelEventArgs): boolean
```

loadingItems イベントを発生させます。

パラメーター

- **e**: **CancelEventArgs** OPTIONAL

継承元 **TreeView**
戻り値 **boolean**

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

lostFocus イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onNodeEditEnded

```
onNodeEditEnded(e: TreeNodeEventArgs): void
```

nodeEditEnded イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

継承元 **TreeView**
戻り値 **void**

onNodeEditEnding

```
onNodeEditEnding(e: TreeNodeEventArgs): boolean
```

nodeEditEnding イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

継承元 **TreeView**
戻り値 **boolean**

onNodeEditStarted

```
onNodeEditStarted(e: TreeNodeEventArgs): void
```

nodeEditStarted イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

継承元 **TreeView**
戻り値 **void**

onNodeEditStarting

```
onNodeEditStarting(e: TreeNodeEventArgs): boolean
```

nodeEditStarting イベントを発生させます。

パラメーター

- **e**: **TreeNodeEventArgs**
  イベントデータを含む**TreeNodeEventArgs** 。

継承元 **TreeView**
戻り値 **boolean**

onRefreshed

```
onRefreshed(e?: EventArgs): void
```

refreshed .イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onRefreshing

```
onRefreshing(e?: EventArgs): void
```

refreshing.イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **Control**
戻り値 **void**

onSelectedItemChanged

```
onSelectedItemChanged(e?: EventArgs): void
```

selectedItemChanged イベントを発生させます。

パラメーター

- **e**: **EventArgs** OPTIONAL

継承元 **TreeView**
戻り値 **void**

refresh

```
refresh(fullUpdate?: boolean): void
```

ツリーを再作成するためにオーバーライドされます。

パラメーター

- **fullUpdate**: **boolean** OPTIONAL
  内容だけでなくコントロールのレイアウトも更新するかどうかを示します。

継承元 **TreeView**
戻り値 **void**

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

HTML要素で存在するすべてのWijmoコントロールを更新する。

コントロールが時間おいて更新される代わりに直ちに更新されること以外は**invalidateAll** メソッドと同様です。

パラメーター

- **e**: **HTMLElement** OPTIONAL
  コンテナー要素。 nullに設定すると、ページ上に存在するすべてのWijmoコントロールが無効化されます。

継承元 **Control**
戻り値 **void**

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

この**Control** が所有する要素にアタッチされている1つまたは複数のイベントリスナーを解除します。

パラメーター

- **target**: **EventTarget** OPTIONAL
  イベントのターゲット要素。 nullの場合、すべてのターゲットにアタッチされているリスナーが解除されます。
- **type**: **string** OPTIONAL
  イベントを指定する文字列。 nullの場合、すべてのイベントにアタッチされているリスナーが解除されます。
- **fn**: **any** OPTIONAL
  削除するハンドラ。 nullの場合は、すべてのハンドラが削除されます。
- **capture**: **boolean** OPTIONAL
  リスナーがキャプチャリスナーかどうか。 nullの場合、キャプチャリスナーと非キャプチャリスナーの両方が解除されます。

継承元 **Control**
戻り値 **number**

startEditing

```
startEditing(node?: TreeNode): boolean
```

特定の**TreeNode** の編集を開始します。

パラメーター

- **node**: **TreeNode** OPTIONAL
  編集する**TreeNode**。 指定しない場合は、現在選択されているノードが使用されます。

継承元 **TreeView**
戻り値 **boolean**

イベント

checkedItemsChanged

checkedItems プロパティの値が変化すると発生します。

継承元 **TreeView**
引数 **EventArgs**

dragEnd

マウスまたはキーボードでノードを別の位置にドロップするか、操作をキャンセルして、 ドラッグ/ドロップ操作を終了すると発生します。

継承元 **TreeView**
引数 **EventArgs**

dragOver

ユーザーが**TreeView** で1つのノードを他のノードの上にドラッグしているときに発生します。

このイベントは、**allowDragging** プロパティがtrueに設定されている場合にのみ発生します。

イベントの**cancel**パラメータをtrueに設定することで、 特定のノードや位置の上にドロップできないようにすることができます。

継承元　　**TreeView**
引数　　**TreeNodeDragDropEventArgs**

dragStart

ユーザーが行のドラッグを開始するときに発生します。

このイベントは、**allowDragging** プロパティがtrueに設定されている場合にのみ発生します。

イベントの**cancel**パラメータをtrueに設定することで、 ノードをドラッグできないようにすることができます。

継承元　　**TreeView**
引数　　**TreeNodeEventArgs**

drop

ユーザーが**TreeView** でノードをドロップしたときに発生します。

継承元　　**TreeView**
引数　　**TreeNodeDragDropEventArgs**

formatItem

ノードを表す要素が作成されたときに発生します。

このイベントを使用して、表示するノードを書式設定できます。

次の例は、**formatItem**イベントを使用して、ツリーの新しい項目の右側に "new"バッジを追加します。

```
var treeViewFmtItem = new wijmo.input.TreeView('#treeViewFmtItem', {
    displayMemberPath: 'header',
    childItemsPath: 'items',
    itemsSource: items,
    formatItem: function (s, e) {
        if (e.dataItem.newItem) {
            e.element.innerHTML +=
                '<img style="margin-left:6px" src="resources/new.png"/>';
        }
    }
});
```

継承元　　**TreeView**
引数　　**FormatNodeEventArgs**

gotFocus

コントロールがフォーカスを取得したときに発生します。

継承元　　**Control**
引数　　**EventArgs**

isCheckedChanged

isChecked プロパティの値が変化した後に発生します。

継承元 **TreeView**
引数 **TreeNodeEventArgs**

isCheckedChanging

isChecked プロパティの値が変化する前に発生します。

継承元 **TreeView**
引数 **TreeNodeEventArgs**

isCollapsedChanged

isCollapsed プロパティの値が変化した後に発生します。

継承元 **TreeView**
引数 **TreeNodeEventArgs**

isCollapsedChanging

isCollapsed プロパティの値が変化する前に発生します。

継承元 **TreeView**
引数 **TreeNodeEventArgs**

itemClicked

ユーザーが項目をクリックするか、［Enter］キーを押して、項目が選択されたときに発生します。

通常、このイベントはナビゲーションツリーで使用されます。 **selectedItem** プロパティを使用して、 クリックされた項目を取得します。

継承元 **TreeView**
引数 **EventArgs**

itemsSourceChanged

itemsSource プロパティの値が変化すると発生します。

継承元 **TreeView**
引数 **EventArgs**

loadedItems

ツリー項目が生成された後に発生します。

継承元 **TreeView**
引数 **EventArgs**

loadingItems

ツリー項目が生成される前に発生します。

継承元 **TreeView**
引数 **CancelEventArgs**

lostFocus

コントロールがフォーカスを失ったときに発生します。

継承元 **Control**
引数 **EventArgs**

nodeEditEnded

TreeNode が編集モードを出た後に発生します。

継承元 **TreeView**
引数 **TreeNodeEventArgs**

nodeEditEnding

TreeNode が編集モードを出る前に発生します。

継承元 **TreeView**
引数 **TreeNodeEventArgs**

nodeEditStarted

TreeNode が編集モードに入った後に発生します。

継承元 **TreeView**
引数 **TreeNodeEventArgs**

nodeEditStarting

TreeNode が編集モードに入る前に発生します。

継承元 **TreeView**
引数 **TreeNodeEventArgs**

refreshed

コントロールが内容を更新した後で発生します。

継承元 **Control**
引数 **EventArgs**

refreshing

コントロールが内容を更新する直前に発生します。

継承元 **Control**
引数 **EventArgs**

selectedItemChanged

selectedItem プロパティの値が変化すると発生します。

継承元 **TreeView**
引数 **EventArgs**

wijmo.react モジュール

ファイル `wijmo.react.js`
モジュール **wijmo.react**

React用のWijmo相互運用モジュール。

このモジュールは、Wijmoコントロールをカプセル化したReactコンポーネントを提供します。

これを使用するには、ReactライブラリとReactDOMライブラリへの参照、および通常のWijmo CSSファイルとjsファイルをアプリケーションに含める必要があります。

ReactコンポーネントにWijmoコントロールを追加するには、JSX（またはTSX）ファイルに適切なタグを含めます。たとえば、次のコードは、Reactコンポーネントに**InputNumber** コントロールを追加します。

```
<label htmlFor="inputnumber">InputNumberコントロールの例：</label>
<Wj.InputNumber
  id="inputNumber"
  format="c2"
  min={ 0 } max={ 10 } step={ .5 }
  value={ this.state.value }
  valueChanged={ this.valueChanged }/>
```

この例は、次の重要なポイントを示しています。

1. Wijmoコントロールのタグ名は、"Wj"プリフィックスで始まり、その後にコントロール名が続きます。"Wj"は、完全なモジュール名"wijmo.react"の省略形です。完全なモジュール名も使用できます。
2. タグ属性名は、コントロールのプロパティやイベントと同じです。
3. 引用符で囲まれた属性値は文字列として解釈され、中かっこで囲まれた値はJavaScript式として解釈されます。
4. Reactコンポーネントには暗黙の双方向連結がありません。したがって、値を変更するコントロールは、通常、イベントハンドラを使用して明示的に親コンポーネントの状態に変更を適用します。

この最後のポイントの具体例として、上記のマークアップを含むコンポーネントには、通常、次のような"valueChanged"イベントハンドラが実装されます。

```
valueChanged: function(s, e) {
  this.setState({ value, s.value });
}
```

このイベントハンドラは、Reactの**setState**メソッドを使用してコンポーネントの状態を更新することで、UIの更新を自動的にトリガします。 このメソッドは、"state"オブジェクトに直接書き込みを行わないことに注意してください。Reactアプリケーション内では、これを不変オブジェクトとして扱う必要があります。

すべてのWijmo Reactコンポーネントには、"initialized"イベントがあります。これは、コントロールがページに追加されて初期化された後に発生します。このイベントを使用して、マークアップでプロパティを設定するほかに、 追加的な初期化を実行できます。次に例を示します。

```
<Wj.FlexGrid
  initialized={ function(s,e) {

    // グリッドにカスタムMergeManagerを割り当てます
    s.mergeManager = new CustomMergeManager(s);
  }}

/>
```

クラス

- AutoComplete
- BulletGraph
- Calendar
- ColorPicker
- ComboBox
- ComponentBase
- FinancialChart

- FinancialChartSeries
- FlexChart
- FlexChartAnimation
- FlexChartAnnotation
- FlexChartAnnotationLayer
- FlexChartAtr
- FlexChartAxis
- FlexChartBollingerBands
- FlexChartBoxWhisker
- FlexChartCci
- FlexChartDataLabel
- FlexChartDataPoint
- FlexChartEnvelopes
- FlexChartErrorBar
- FlexChartFibonacci
- FlexChartFibonacciArcs
- FlexChartFibonacciFans
- FlexChartFibonacciTimeZones
- FlexChartGestures
- FlexChartLegend
- FlexChartLineMarker
- FlexChartMacd
- FlexChartMacdHistogram
- FlexChartMovingAverage
- FlexChartParametricFunctionSeries
- FlexChartPlotArea
- FlexChartRangeSelector
- FlexChartRsi
- FlexChartSeries
- FlexChartStochastic
- FlexChartTrendLine
- FlexChartWaterfall
- FlexChartWilliamsR
- FlexChartYFunctionSeries
- FlexGrid
- FlexGridColumn
- FlexGridFilter
- FlexPie
- FlexPieDataLabel
- FlexRadar
- FlexRadarAxis
- FlexRadarSeries
- FlexSheet
- GroupPanel
- InputColor
- InputDate
- InputDateTime
- InputMask
- InputNumber

- InputTime
- LinearGauge
- ListBox
- Menu
- MultiAutoComplete
- MultiRow
- MultiSelect
- PdfViewer
- PivotChart
- PivotGrid
- PivotPanel
- Popup
- RadialGauge
- Range
- ReportViewer
- Sheet
- Slicer
- Sunburst
- TreeMap
- TreeView

AutoComplete クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

AutoComplete コントロールをカプセル化するReactコンポーネント。

BulletGraph クラス

ファイル wijmo.react.js
モジュール **wijmo.react**
基本クラス **ComponentBase**
表示 ☐継承されたメンバー ☐イベント発生元

BulletGraph コントロールをカプセル化するReactコンポーネント。

次の例は、JSXで**BulletGraph** コントロールをインスタンス化して初期化する方法を示します。

```
<Wj.BulletGraph
  min={ 0 } max={ 1000 } step={ 50 } isReadOnly={ false }
  value={ this.state.view.currentItem.sales }
  valueChanged={ this.salesChanged }
  format="c0" thumbSize={ 20 } showRanges={ false }
  face={{ thickness:0.5 }}
  pointer={{ thickness:0.5 }}
  ranges={[
      { min: 0, max: 333, color: 'red' },
      { min: 333, max: 666, color: 'gold' },
      { min: 666, max: 1000, color: 'green' }
  ]} />
```

このコードは、**min**、**max**、**step**、**isReadOnly**の各プロパティを設定して、 ゲージの範囲を定義し、ユーザーがゲージの値を編集できるようにします。 次に、**value**プロパティと**valueChanged**プロパティを設定して、ゲージの値に対して双方向連結を構築します。

次に、**format**、**thumbSize**、**showRanges**の各プロパティを設定して、ゲージの外観を定義します。 最後に、このマークアップは、 **face**範囲と**pointer**範囲の太さを設定し、 ゲージの現在の値に応じて**value**範囲の色を制御するために、いくつかの範囲を設定します。

Calendar クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐ 継承されたメンバー ☐ イベント発生元

Calendar コントロールをカプセル化するReactコンポーネント。

ColorPicker クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐ 継承されたメンバー ☐ イベント発生元

ColorPicker コントロールをカプセル化するReactコンポーネント。

ComboBox クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

ComboBox コントロールをカプセル化するReactコンポーネント。

ComponentBase クラス

ファイル wijmo.react.js
モジュール **wijmo.react**
派生クラス **AutoComplete**, **BulletGraph**, **Calendar**, **ColorPicker**, **ComboBox**, **FinancialChart**, **FinancialChartSeries**, **FlexChart**, **FlexChartAnimation**, **FlexChartAnnotation**, **FlexChartAnnotationLayer**, **FlexChartAtr**, **FlexChartAxis**, **FlexChartBollingerBands**, **FlexChartBoxWhisker**, **FlexChartCci**, **FlexChartDataLabel**, **FlexChartDataPoint**, **FlexChartEnvelopes**, **FlexChartErrorBar**, **FlexChartFibonacci**, **FlexChartFibonacciArcs**, **FlexChartFibonacciFans**, **FlexChartFibonacciTimeZones**, **FlexChartGestures**, **FlexChartLegend**, **FlexChartLineMarker**, **FlexChartMacd**, **FlexChartMacdHistogram**, **FlexChartMovingAverage**, **FlexChartParametricFunctionSeries**, **FlexChartPlotArea**, **FlexChartRangeSelector**, **FlexChartRsi**, **FlexChartSeries**, **FlexChartStochastic**, **FlexChartTrendLine**, **FlexChartWaterfall**, **FlexChartWilliamsR**, **FlexChartYFunctionSeries**, **FlexGrid**, **FlexGridColumn**, **FlexGridFilter**, **FlexPie**, **FlexPieDataLabel**, **FlexRadar**, **FlexRadarAxis**, **FlexRadarSeries**, **FlexSheet**, **GroupPanel**, **InputColor**, **InputDate**, **InputDateTime**, **InputMask**, **InputNumber**, **InputTime**, **LinearGauge**, **ListBox**, **Menu**, **MultiAutoComplete**, **MultiRow**, **MultiSelect**, **PdfViewer**, **PivotChart**, **PivotGrid**, **PivotPanel**, **Popup**, **RadialGauge**, **Range**, **ReportViewer**, **Sheet**, **Slicer**, **Sunburst**, **TreeMap**, **TreeView**
表示 ☐継承されたメンバー ☐イベント発生元

ReactのすべてのWijmoコンポーネントの基本クラス。

FinancialChart クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐ 継承されたメンバー ☐ イベント発生元

FinancialChart コントロールをカプセル化するReactコンポーネント。

FinancialChartSeries クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**FinancialSeries** を表す Reactコンポーネント。

FlexChart クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart コントロールをカプセル化するReactコンポーネント。

次の例は、JSXで**FlexChart** コントロールをインスタンス化して初期化する方法を示します。

```
<Wj.FlexChart
  itemsSource={ this.state.data }
  bindingX="name"
  header={ this.state.header }
  footer={ this.state.footer }
  axisX={{ title: this.state.titleX }}
  axisY={{ title: this.state.titleY }}
  legend={{ position: this.state.legendPosition }}
  series={[
      { name: 'Sales', binding: 'sales' },
      { name: 'Expenses', binding: 'expenses' },
      { name: 'Downloads', binding: 'downloads', chartType: 'LineSymbols' }
  ]} />
```

このコードは、チャート化するデータを含むコレクションを**itemsSource**プロパティに設定し、 チャートのX値として使用するデータプロパティの名前を**bindingX**プロパティに設定します。

また、**header**プロパティと**footer**プロパティを設定してチャートタイトルを指定し、 チャートの軸と凡例をカスタマイズします。

最後に、チャートに表示するデータ項目を指定する配列を**series**プロパティに設定します。

FlexChartAnimation クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**ChartAnimation** を表すReactコンポーネント。 **FlexChart** 、**FlexPie** 、**FinancialChart**、 または**FlexRadar**。

FlexChartAnnotation クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexChartAnnotationLayer コントロールの**annotation** を表す Reactコンポーネント。

FlexChartAnnotationLayer クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**AnnotationLayer** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartAtr クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**ATR** を表す Reactコンポーネント。

FlexChartAxis クラス

ファイル wijmo.react.js
モジュール **wijmo.react**
基本クラス **ComponentBase**
表示 ☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**Axis** を表すReactコンポーネント。 **FlexChart** 、**FlexChartSeries** 、**FinancialChart**、 または**FinancialChartSeries**。

FlexChartBollingerBands クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**BollingerBands** を表す Reactコンポーネント。

FlexChartBoxWhisker クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**BoxWhisker** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartCci クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**CCI** を表す Reactコンポーネント。

FlexChartDataLabel クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart コントロールの**DataLabel** を表す Reactコンポーネント。

FlexChartDataPoint クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexChartAnnotation コントロールの**DataPoint** を表す Reactコンポーネント。

FlexChartEnvelopes クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**Envelopes** を表す Reactコンポーネント。

FlexChartErrorBar クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart コントロールの**ErrorBar** を表す Reactコンポーネント。

FlexChartFibonacci クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**Fibonacci** を表す Reactコンポーネント。

FlexChartFibonacciArcs クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**FibonacciArcs** を表す Reactコンポーネント。

FlexChartFibonacciFans クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**FibonacciFans** を表す Reactコンポーネント。

FlexChartFibonacciTimeZones クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**FibonacciTimeZones** を表す Reactコンポーネント。

FlexChartGestures クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**ChartGestures** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartLegend クラス

ファイル wijmo.react.js
モジュール **wijmo.react**
基本クラス **ComponentBase**
表示 ☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**Legend** を表すReactコンポーネント。 **FlexChart** 、**FlexPie** 、**FinancialChart**、 **FlexRadar**、 または**Sunburst**。

FlexChartLineMarker クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**LineMarker** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartMacd クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**Macd** を表す Reactコンポーネント。

FlexChartMacdHistogram クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**MacdHistogram** を表す Reactコンポーネント。

FlexChartMovingAverage クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**MovingAverage** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartParametricFunctionSeries クラス

ファイル wijmo.react.js
モジュール **wijmo.react**
基本クラス **ComponentBase**
表示 ☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**ParametricFunctionSeries** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartPlotArea クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**PlotArea** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartRangeSelector クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**RangeSelector** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartRsi クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**RSI** を表す Reactコンポーネント。

FlexChartSeries クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexChart コントロールの**Series** を表す Reactコンポーネント。

FlexChartStochastic クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**Stochastic** を表す Reactコンポーネント。

FlexChartTrendLine クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐ 継承されたメンバー ☐ イベント発生元

次のコンポーネントのいずれかに**TrendLine** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartWaterfall クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**Waterfall** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexChartWilliamsR クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FinancialChart コントロールの**WilliamsR** を表す Reactコンポーネント。

FlexChartYFunctionSeries クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**YFunctionSeries** を表すReactコンポーネント。 **FlexChart** 、または**FinancialChart**。

FlexGrid クラス

ファイル wijmo.react.js
モジュール **wijmo.react**
基本クラス **ComponentBase**
表示 ☐継承されたメンバー ☐イベント発生元

FlexGrid コントロールをカプセル化するReactコンポーネント。

次の例は、JSXで**FlexGrid** コントロールをインスタンス化して初期化する方法を示します。

```
<Wj.FlexGrid
  autoGenerateColumns={ false }
  columns={[
    { binding: 'name', header: 'Name' },
    { binding: 'sales', header: 'Sales', format: 'c0' },
    { binding: 'expenses', header: 'Expenses', format: 'c0' },
    { binding: 'active', header: 'Active' },
    { binding: 'date', header: 'Date' }
  ]}
  itemsSource={ this.state.data } />
```

このコードは、**autoGenerateColumns**プロパティをfalseに設定してから、**columns**プロパティを設定し、 最後に**itemsSource**プロパティを設定します。 この順序は重要です。 この順序によってグリッドが自動的に列を生成することを防ぎます。

FlexGridColumn クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexGrid コントロールの**Column** を表す Reactコンポーネント。

FlexGridFilter クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexGrid コントロールの**FlexGridFilter** を表す Reactコンポーネント。

FlexPie クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexPie コントロールをカプセル化するReactコンポーネント。

FlexPieDataLabel クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexPie コントロールの**PieDataLabel** を表す Reactコンポーネント。

FlexRadar クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexRadar コントロールをカプセル化するReactコンポーネント。

FlexRadarAxis クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**FlexRadarAxis** を表すReactコンポーネント。 **FlexRadar** 、または**FlexRadarSeries**。

FlexRadarSeries クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexRadar コントロールの**FlexRadarSeries** を表す Reactコンポーネント。

FlexSheet クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexSheet コントロールをカプセル化するReactコンポーネント。

GroupPanel クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

GroupPanel コントロールをカプセル化するReactコンポーネント。

InputColor クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

InputColor コントロールをカプセル化するReactコンポーネント。

InputDate クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

InputDate コントロールをカプセル化するReactコンポーネント。

InputDateTime クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

InputDateTime コントロールをカプセル化するReactコンポーネント。

InputMask クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

InputMask コントロールをカプセル化するReactコンポーネント。

InputNumber クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐ 継承されたメンバー ☐ イベント発生元

InputNumber コントロールをカプセル化するReactコンポーネント。

InputTime クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

InputTime コントロールをカプセル化するReactコンポーネント。

LinearGauge クラス

ファイル wijmo.react.js
モジュール **wijmo.react**
基本クラス **ComponentBase**
表示 ☐継承されたメンバー ☐イベント発生元

LinearGauge コントロールをカプセル化するReactコンポーネント。

次の例は、JSXで**LinearGauge** コントロールをインスタンス化して初期化する方法を示します。

```
<Wj.LinearGauge
  min={ 0 } max={ 1000 } step={ 50 } isReadOnly={ false }
  value={ this.state.view.currentItem.sales }
  valueChanged={ this.salesChanged }
  format="c0" thumbSize={ 20 } showRanges={ false }
  face={{ thickness:0.5 }}
  pointer={{ thickness:0.5 }}
  ranges={[
      { min: 0, max: 333, color: 'red' },
      { min: 333, max: 666, color: 'gold' },
      { min: 666, max: 1000, color: 'green' }
  ]} />
```

このコードは、**min**、**max**、**step**、**isReadOnly**の各プロパティを設定して、 ゲージの範囲を定義し、ユーザーがゲージの値を編集できるようにします。 次に、**value**プロパティと**valueChanged**プロパティを設定して、ゲージの値に対して双方向連結を構築します。

次に、**format**、**thumbSize**、**showRanges**の各プロパティを設定して、ゲージの外観を定義します。 最後に、このマークアップは、 **face**範囲と**pointer**範囲の太さを設定し、 ゲージの現在の値に応じて**value**範囲の色を制御するために、いくつかの範囲を設定します。

ListBox クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

ListBox コントロールをカプセル化するReactコンポーネント。

Menu クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

Menu コントロールをカプセル化するReactコンポーネント。

MultiAutoComplete クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

MultiAutoComplete コントロールをカプセル化するReactコンポーネント。

MultiRow クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

MultiRow コントロールをカプセル化するReactコンポーネント。

MultiSelect クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

MultiSelect コントロールをカプセル化するReactコンポーネント。

PdfViewer クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

PdfViewer コントロールをカプセル化するReactコンポーネント。

PivotChart クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

PivotChart コントロールをカプセル化するReactコンポーネント。

PivotGrid クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

PivotGrid コントロールをカプセル化するReactコンポーネント。

PivotPanel クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

PivotPanel コントロールをカプセル化するReactコンポーネント。

Popup クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐ 継承されたメンバー ☐ イベント発生元

Popup コントロールをカプセル化するReactコンポーネント。

RadialGauge クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

RadialGauge コントロールをカプセル化するReactコンポーネント。

次の例は、JSXで**RadialGauge** コントロールをインスタンス化して初期化する方法を示します。

```
<Wj.RadialGauge
  min={ 0 } max={ 1000 } step={ 50 } isReadOnly={ false }
  value={ this.state.view.currentItem.sales }
  valueChanged={ this.salesChanged }
  format="c0" thumbSize={ 20 } showRanges={ false }
  face={{ thickness:0.5 }}
  pointer={{ thickness:0.5 }}
  ranges={[
      { min: 0, max: 333, color: 'red' },
      { min: 333, max: 666, color: 'gold' },
      { min: 666, max: 1000, color: 'green' }
  ]} />
```

このコードは、**min**、**max**、**step**、**isReadOnly**の各プロパティを設定して、 ゲージの範囲を定義し、ユーザーがゲージの値を編集できるようにします。 次に、**value**プロパティと**valueChanged**プロパティを設定して、ゲージの値に対して双方向連結を構築します。

次に、**format**、**thumbSize**、**showRanges**の各プロパティを設定して、ゲージの外観を定義します。 最後に、このマークアップは、 **face**範囲と**pointer**範囲の太さを設定し、 ゲージの現在の値に応じて**value**範囲の色を制御するために、いくつかの範囲を設定します。

Range クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

次のコンポーネントのいずれかに**Range** を表すReact コンポーネント。 **LinearGauge** 、**BulletGraph**、 または**RadialGauge**。

ReportViewer クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐ 継承されたメンバー ☐ イベント発生元

ReportViewer コントロールをカプセル化するReactコンポーネント。

Sheet クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

FlexSheet コントロールの**Sheet** を表す Reactコンポーネント。

Slicer クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

Slicer コントロールをカプセル化するReactコンポーネント。

Sunburst クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐継承されたメンバー ☐イベント発生元

Sunburst コントロールをカプセル化するReactコンポーネント。

TreeMap クラス

ファイル	`wijmo.react.js`
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐ 継承されたメンバー ☐ イベント発生元

TreeMap コントロールをカプセル化するReactコンポーネント。

TreeView クラス

ファイル	wijmo.react.js
モジュール	**wijmo.react**
基本クラス	**ComponentBase**
表示	☐ 継承されたメンバー ☐ イベント発生元

TreeView コントロールをカプセル化するReactコンポーネント。

wijmo.vue モジュール

ファイル `wijmo.vue.js`
モジュール **wijmo.vue**

Vue用のWijmo相互運用モジュール。

このモジュールは、Wijmoコントロールをカプセル化したVueコンポーネントを提供します。

これを使用するには、Vueフレームワークへの参照、 および通常のWijmo CSSファイルとjsファイルをアプリケーションに含める必要があります。

VueページにWijmoコントロールを追加するには、HTMLファイルに適切なタグを含めます。たとえば、次のコードは、 Vueページに**InputNumber** コントロールを追加します。

```
<wj-input-number
  format="c2"
  placeholder="Sales"
  :value.sync="view.currentItem.sales"
  :value-changed="refreshView"
  :min="0"
  :max="10000"
  :step="100"
  :is-required="false">
</wj-input-number>
```

この例は、次の重要なポイントを示しています。

1. Wijmoコントロールのタグ名は、"wj"プリフィックスで始まり、その後に 英小文字とハイフン区切り文字を使用したコントロール名が続きます。
2. タグ属性名は、コントロールのプロパティやイベントと同じです。
3. 属性名の前のコロンは、属性値がJavaScript式として解釈されることを示します （ `:min="0"` など）。
4. イベントハンドラは、通常のプロパティと同様に指定します （ `:value-changed="refreshView"` など）。
5. 属性名の後の `.sync` サフィックスは、双方向連結を作成します （ `:value.sync="view.currentItem.sales"` など）。

すべてのWijmo Vueコンポーネントには、"initialized"イベントがあります。 これは、コントロールがページに追加されて初期化された後に発生します。 このイベントを使用して、マークアップでプロパティを設定するほかに、 追加的な初期化を実行できます。次に例を示します。

```
<wj-flex-grid :initialized="initGrid">
</wj-flex-grid>
```

```
// Vueアプリケーション
var app = new Vue({
  el: '#app',
  methods: {
    initGrid: function(s, e) {

      // グリッドにカスタムMergeManagerを割り当てます
      s.mergeManager = new CustomMergeManager(s);
    }
  }
});
```

すべてのWijmo Vueコンポーネントには、マークアップでコントロールのプロパティを使用するための "control"疑似プロパティが含まれます。次に例を示します。

```
<wj-list-box style="height:150px;width:250px;"
  :items-source="countries"
  control="listBox">
</wj-list-box>
<p>
  selectedIndex: {{ listBox.selectedIndex }}</p>
<p>
  selectedValue: {{ listBox.selectedValue }}</p>
```

このマークアップは、コントローラーの**listBox**プロパティを **ListBox**コントロールのインスタンスに設定して、それをマークアップで使用できるようにします。

これを含むVueコンテキストの**data**オブジェクトで**listBox**プロパティを 定義することが重要です。そうしないと、コンポーネントのプロパティが 適切に更新されません。次に例を示します。

```
// Vueアプリケーション
var app = new Vue({
  el: '#app',
  data: {
    listBox: null // ListBoxコントロールへの参照を受け取ります
  }
});
```

プロパティ

- WjAutoComplete
- WjBulletGraph
- WjCalendar
- WjColorPicker
- WjComboBox
- WjFlexChart
- WjFlexChartAxis
- WjFlexChartLegend
- WjFlexChartSeries
- WjFlexGrid
- WjFlexGridColumn
- WjFlexGridFilter
- WjFormat
- WjInclude
- WjInputColor
- WjInputDate
- WjInputDateTime
- WjInputMask
- WjInputNumber
- WjInputTime
- WjLinearGauge
- WjListBox
- WjMenu
- WjMultiAutoComplete
- WjMultiSelect
- WjPivotChart
- WjPivotGrid
- WjPivotPanel
- WjPopup
- WjRadialGauge
- WjRange
- WjSlicer
- WjTreeView

プロパティ

- WjAutoComplete

AutoComplete コントロールをカプセル化するVueコンポーネント。

型

- WjBulletGraph

BulletGraph コントロールをカプセル化するVueコンポーネント。

型

- WjCalendar

Calendar コントロールをカプセル化するVueコンポーネント。

型

- WjColorPicker

ColorPicker コントロールをカプセル化するVueコンポーネント。

型

- WjComboBox

ComboBox コントロールをカプセル化するVueコンポーネント。

型

- WjFlexChart

FlexChart コントロールをカプセル化するVueコンポーネント。

次の例は、Vueマークアップで**FlexChart** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-flex-chart
    :items-source="data"
    binding-x="country"
    :header="props.header"
    :footer="props.footer">

    <wj-flex-chart-legend :position="props.legendPosition">
    </wj-flex-chart-legend>
    <wj-flex-chart-axis wj-property="axisX" :title="props.titleX">
    </wj-flex-chart-axis>
    <wj-flex-chart-axis wj-property="axisY" :title="props.titleY">
    </wj-flex-chart-axis>

    <wj-flex-chart-series name="Sales" binding="sales">
    </wj-flex-chart-series>
    <wj-flex-chart-series name="Expenses" binding="expenses">
    </wj-flex-chart-series>
    <wj-flex-chart-series name="Downloads" binding="downloads">
    </wj-flex-chart-series>
</wj-flex-chart>
```

このコードは、チャートデータを含むコレクションを**itemsSource**プロパティに設定し、 チャートのX値を含むデータプロパティを**bindingX**プロパティに設定します。 また、チャートの**header**プロパティと**footer**プロパティを設定して、 チャートの上下に表示するタイトルを定義します。

wj-flex-chart-legendコンポーネントと**wj-flex-chart-axis**コンポーネントは、 チャートの凡例と軸をカスタマイズするために使用されます。

最後に、3つの**wj-flex-chart-series**コンポーネントを使用して、 チャートに表示するデータプロパティを指定します。

型

- WjFlexChartAxis

FlexChart コントロールの**Axis** を表すVueコンポーネント。

型

- WjFlexChartLegend

FlexChart コントロールの**Legend** を表す Vueコンポーネント。

型

- WjFlexChartSeries

FlexChart コントロールの**Series** を表すVueコンポーネント。

型

- WjFlexGrid

FlexGrid コントロールをカプセル化するVueコンポーネント。

次の例は、Vueマークアップを使用して、 **FlexGrid** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-flex-grid
  :items-source="data">
  <wj-flex-grid-column binding="name" header="Name">
  </wj-flex-grid-column>
  <wj-flex-grid-column binding="sales" header="Sales" format="c0">
  </wj-flex-grid-column>
  <wj-flex-grid-column binding="expenses" header="Expenses" format="c0">
  </wj-flex-grid-column>
  <wj-flex-grid-column binding="active" header="Active">
  </wj-flex-grid-column>
  <wj-flex-grid-column binding="date" header="Date">
  </wj-flex-grid-column>
</wj-flex-grid>
```

このコードは、グリッドデータを含むコレクションを**itemsSource**プロパティに設定してから、 **wj-flex-grid-column**コンポーネントを使用して、表示する列を指定します。

型

- WjFlexGridColumn

FlexGrid コントロールの**Column** を表す Vueコンポーネント。

型

- WjFlexGridFilter

FlexGrid コントロールの**FlexGridFilter** を表す Vueコンポーネント。 次の例は、Vueマークアップを使用して、 フィルタ付きの **FlexGrid** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-flex-grid
  :items-source="data">
  <wj-flex-grid-filter></wj-flex-grid-filter>
</wj-flex-grid>
```

型

- WjFormat

グローバル化された書式設定を日付や数値に適用するVueフィルタ。

たとえば、次のコードは、**wj-format**フィルタを使用して、 現在のWijmoカルチャに基づいて数値を通貨値として書式設定し、日付を短い日付として書式設定します。

```
<p>value: {{ theAmount | wj-format 'c' }}</p>
<p>date: {{ theDate | wj-format 'd' }}</p>
```

型

- WjInclude

特定のHTMLフラグメントをドキュメントに含めるVueコンポーネント。

wj-includeコンポーネントは、ドキュメントにロードおよび挿入するファイルを指定する**src**属性を受け取ります。次に例を示します。

```
<wj-popup control="modalDialog" :modal="true" :hide-trigger="None">
  <wj-include src="includes/dialog.htm"></wj-include>
</wj-popup>
```

型

- WjInputColor

InputColor コントロールをカプセル化するVueコンポーネント。

型

- WjInputDate

InputDate コントロールをカプセル化するVueコンポーネント。

型

- WjInputDateTime

InputDateTime コントロールをカプセル化するVueコンポーネント。

型

- WjInputMask

InputMask コントロールをカプセル化するVueコンポーネント。

型

- WjInputNumber

InputNumber コントロールをカプセル化するVueコンポーネント。

型

- WjInputTime

InputTime コントロールをカプセル化するVueコンポーネント。

型

- WjLinearGauge

LinearGauge コントロールをカプセル化するVueコンポーネント。

次の例は、Vueマークアップで**LinearGauge** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-linear-gauge
    :min="0" :max="1000" :step="50" :is-read-only="false"
    format="c0" :thumb-size="20"
    :show-ranges="false"
    :value.sync="sales">
    <wj-range wj-property="face" :thickness="0.5">
    </wj-range>
    <wj-range wj-property="pointer" :thickness="0.5">
    </wj-range>
    <wj-range :min="0" :max="333" color="red">
    </wj-range>
    <wj-range :min="333" :max="666" color="gold">
    </wj-range>
    <wj-range :min="666" :max="1000" color="green">
    </wj-range>
</wj-linear-gauge>
```

このコードは、**min**、**max**、**step**、 **isReadOnly**の各プロパティを設定して、ゲージの範囲を定義し、ユーザーがゲージの値を編集できるようにします。 次に、ゲージの**value**プロパティをコントローラーの**sales**変数に連結します。

次に、**format**、**thumbSize**、**showRanges**の各プロパティを設定して、ゲージの外観を定義します。 最後に、このマークアップは、 **face**範囲と**pointer**範囲の太さを設定し、 ゲージの現在の値に応じて**value**範囲の色を制御するために、いくつかの範囲を設定します。

型

- WjListBox

ListBox コントロールをカプセル化するVueコンポーネント。

型

- WjMenu

Menu コントロールをカプセル化するVueコンポーネント。

型

- WjMultiAutoComplete

MultiAutoComplete コントロールをカプセル化するVueコンポーネント。

型

- WjMultiSelect

MultiSelect コントロールをカプセル化するVueコンポーネント。

型

- WjPivotChart

PivotChart コントロールをカプセル化するVueコンポーネント。

型

- WjPivotGrid

PivotGrid コントロールをカプセル化するVueコンポーネント。

型

- WjPivotPanel

PivotPanel コントロールをカプセル化するVueコンポーネント。

型

- WjPopup

Popup コントロールをカプセル化するVueコンポーネント。

型

- WjRadialGauge

RadialGauge コントロールをカプセル化するVueコンポーネント。

次の例は、Vueマークアップで**RadialGauge** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-radial-gauge
    :min="0" :max="1000" :step="50" :is-read-only="false"
    format="c0" :thumb-size="12" :show-text="Value"
    :show-ranges="false"
    :value.sync="sales">
    <wj-range wj-property="face" :thickness="0.5">
    </wj-range>
    <wj-range wj-property="pointer" :thickness="0.5">
    </wj-range>
    <wj-range :min="0" :max="333" color="red">
    </wj-range>
    <wj-range :min="333" :max="666" color="gold">
    </wj-range>
    <wj-range :min="666" :max="1000" color="green">
    </wj-range>
</wj-radial-gauge>
```

このコードは、**min**、**max**、**step**、**isReadOnly**の各プロパティを設定して、 ゲージの範囲を定義し、ユーザーがゲージの値を編集できるようにします。 次に、ゲージの**value**プロパティをコントローラーの**sales**変数に連結します。

次に、**format**、**thumbSize**、**showRanges**の各プロパティを設定して、ゲージの外観を定義します。 最後に、このマークアップは、 **face**範囲と**pointer**範囲の太さを設定し、 ゲージの現在の値に応じて**value**範囲の色を制御するために、いくつかの範囲を設定します。

型

- WjRange

Gauge コントロールの**Range** を表す Vueコンポーネント。

型

- WjSlicer

Slicer コントロールをカプセル化するVueコンポーネント。

型

- WjTreeView

TreeView コントロールをカプセル化するVueコンポーネント。

型

wijmo.vue2 モジュール

ファイル `wijmo.vue2.js`
モジュール **wijmo.vue2**

Vue 2用のWijmo相互運用モジュール。

このモジュールは、Wijmoコントロールをカプセル化したVue 2コンポーネントを提供します。

これを使用するには、Vue 2フレームワーク（RC6以上）への参照、 および通常のWijmo CSSファイルとjsファイルをアプリケーションに含める必要があります。

VueページにWijmoコントロールを追加するには、 HTMLファイルに適切なタグを含めます。たとえば、次のコードは、 Vueページに**InputNumber** コントロールを追加します。

```
<wj-input-number
  format="c2"
  placeholder="Sales"
  :value="sales"
  :value-changed="salesChanged"
  :min="0"
  :max="10000"
  :step="100"
  :is-required="false">
</wj-input-number>


// Wijmoイベントハンドラ
// InputNumber値と比較する"sales"値を更新します
function salesChanged(sender, eventArgs) {
  this.sales = sender.value;
}
```

この例は、次の重要なポイントを示しています。

1. Wijmoコントロールのタグ名は、"wj"プリフィックスで始まり、その後に 英小文字とハイフン区切り文字を使用したコントロール名が続きます。
2. タグ属性名は、コントロールのプロパティやイベントと同じです。
3. 属性名の前のコロンは、属性値がJavaScript式として解釈されることを示します （ `:min="0"` など）。
4. イベントハンドラは、通常のプロパティと同様に指定します （ `:value-changed="salesChanged"` など）。
5. Vue2では、すべての連結が一方向です。上の例では、"salesChanged" イベントハンドラは、モデル内の"sales"プロパティの値を更新します。 これはVue 1からの変更点です。Vue 1では、どのような属性にでも ".sync"サフィックスを追加することで、双方向連結を作成できます。

すべてのWijmo Vueコンポーネントには、"initialized"イベントがあります。 これは、コントロールがページに追加されて初期化された後に発生します。 このイベントを使用して、マークアップでプロパティを設定するほかに、 追加的な初期化を実行できます。次に例を示します。

```
<wj-flex-grid :initialized="initGrid">
</wj-flex-grid>


// Vueアプリケーション
var app = new Vue({
  el: '#app',
  methods: {
    initGrid: function(s, e) {

      // グリッドにカスタムMergeManagerを割り当てます
      s.mergeManager = new CustomMergeManager(s);
    }
  }
});
```

プロパティ

- WjAutoComplete
- WjBulletGraph
- WjCalendar
- WjColorPicker
- WjComboBox

- WjFinancialChart
- WjFinancialChartSeries
- WjFlexChart
- WjFlexChartAxis
- WjFlexChartDataLabel
- WjFlexChartDataPoint
- WjFlexChartLegend
- WjFlexChartLineMarker
- WjFlexChartPlotArea
- WjFlexChartSeries
- WjFlexGrid
- WjFlexGridColumn
- WjFlexGridDetail
- WjFlexGridFilter
- WjFlexPie
- WjFlexPieDataLabel
- WjFlexRadar
- WjFlexRadarAxis
- WjFlexRadarSeries
- WjFlexSheet
- WjFormat
- WjGroupPanel
- WjInclude
- WjInputColor
- WjInputDate
- WjInputDateTime
- WjInputMask
- WjInputNumber
- WjInputTime
- WjLinearGauge
- WjListBox
- WjMenu
- WjMultiAutoComplete
- WjMultiRow
- WjMultiSelect
- WjPdfViewer
- WjPivotChart
- WjPivotGrid
- WjPivotPanel
- WjPopup
- WjRadialGauge
- WjRange
- WjReportViewer
- WjSheet
- WjSlicer
- WjSunburst
- WjTreeMap
- WjTreeView

プロパティ

- WjAutoComplete

AutoComplete コントロールをカプセル化するVueコンポーネント。

型

- WjBulletGraph

BulletGraph コントロールをカプセル化するVueコンポーネント。

型

- WjCalendar

Calendar コントロールをカプセル化するVueコンポーネント。

型

- WjColorPicker

ColorPicker コントロールをカプセル化するVueコンポーネント。

型

- WjComboBox

ComboBox コントロールをカプセル化するVueコンポーネント。

型

- WjFinancialChart

wijmo.chart.finance.FinancialChart コントロールをカプセル化するVueコンポーネント。

型

- WjFinancialChartSeries

WjFinancialChart の**FinancialSeries** を表す Vueコンポーネント。

型

WjFlexChart

FlexChart コントロールをカプセル化するVueコンポーネント。

次の例は、Vueマークアップを使用して、 **FlexChart** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-flex-chart
    :items-source="data"
    binding-x="country"
    :header="props.header"
    :footer="props.footer">

    <wj-flex-chart-legend :position="props.legendPosition">
    </wj-flex-chart-legend>
    <wj-flex-chart-axis wj-property="axisX" :title="props.titleX">
    </wj-flex-chart-axis>
    <wj-flex-chart-axis wj-property="axisY" :title="props.titleY">
    </wj-flex-chart-axis>

    <wj-flex-chart-series name="Sales" binding="sales">
    </wj-flex-chart-series>
    <wj-flex-chart-series name="Expenses" binding="expenses">
    </wj-flex-chart-series>
    <wj-flex-chart-series name="Downloads" binding="downloads">
    </wj-flex-chart-series>
</wj-flex-chart>
```

このコードは、チャートデータを含むコレクションを**itemsSource**プロパティに設定し、 チャートのX値を含むデータプロパティを**bindingX**プロパティに設定します。 また、チャートの**header**プロパティと**footer**プロパティを設定して、 チャートの上下に表示するタイトルを定義します。

wj-flex-chart-legendコンポーネントと**wj-flex-chart-axis**コンポーネントを使用して、 チャートの凡例と軸をカスタマイズします。

最後に、3つの**wj-flex-chart-series**コンポーネントを使用して、 チャートに表示されるデータプロパティを指定します。

型

WjFlexChartAxis

次のコンポーネントのいずれかに**Axis** を表すVueコンポーネント。 **WjFlexChart** 、**WjFlexChartSeries** 、**WjFinancialChart**、 または**WjFinancialChartSeries**.

型

WjFlexChartDataLabel

WjFlexChart のwijmo.chart.DataLabel を表す Vueコンポーネント。

型

WjFlexChartDataPoint

次のコンポーネントのいずれかにwijmo.chart.DataPoint を表すVueコンポーネント。 **WjFlexChartAnnotationText** 、 **WjFlexChartAnnotationEllipse** 、 **WjFlexChartAnnotationRectangle WjFlexChartAnnotationLine** 、**WjFlexChartAnnotationPolygon** 、**WjFlexChartAnnotationCircle** 、**WjFlexChartAnnotationSquare** 、 または**WjFlexChartAnnotationImage** 。

型

WjFlexChartLegend

次のコンポーネントのいずれかに**Axis** を表すVueコンポーネント。 **WjFlexChart** 、**WjFlexPie** 、**WjFinancialChart** 、**WjFlexRadar**、 または**WjSunburst**.

型

- WjFlexChartLineMarker

次のコンポーネントのいずれかに**LineMarker** を表すVueコンポーネント。 wijmo.vue2.WjFlexChart 、またはwijmo.vue2.WjFinancialChart 。

型

- WjFlexChartPlotArea

次のコンポーネントのいずれかにwijmo.chart.PlotArea を表すVueコンポーネント。 wijmo.vue2.WjFlexChart 、またはwijmo.vue2.WjFinancialChart 。

型

- WjFlexChartSeries

WjFlexChart コントロールの**Series** を表す Vueコンポーネント。

型

- WjFlexGrid

FlexGrid コントロールをカプセル化するVueコンポーネント。 次の例は、Vueマークアップを使用して、 **FlexGrid** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-flex-grid
  :items-source="data">
  <wj-flex-grid-column binding="name" header="Name">
  </wj-flex-grid-column>
  <wj-flex-grid-column binding="sales" header="Sales" format="c0">
  </wj-flex-grid-column>
  <wj-flex-grid-column binding="expenses" header="Expenses" format="c0">
  </wj-flex-grid-column>
  <wj-flex-grid-column binding="active" header="Active">
  </wj-flex-grid-column>
  <wj-flex-grid-column binding="date" header="Date">
  </wj-flex-grid-column>
</wj-flex-grid>
```

このコードは、グリッドデータを含むコレクションを**itemsSource**プロパティに設定してから、 **wj-flex-grid-column**コンポーネントを使用して、表示する列を指定します。

型

- WjFlexGridColumn

FlexGrid コントロールの**Column** を表す Vueコンポーネント。

型

- WjFlexGridDetail

WjFlexGrid の**FlexGridDetailProvider** を表す Vueコンポーネント。

型

- WjFlexGridFilter

WjFlexGrid コントロールの**FlexGridFilter** を表す Vueコンポーネント。

次の例は、Vueマークアップを使用して、 フィルタ付きの **FlexGridFilter** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-flex-grid
  :items-source="data">
  <wj-flex-grid-filter></wj-flex-grid-filter>
</wj-flex-grid>
```

型

- WjFlexPie

wijmo.chart.FlexPie コントロールをカプセル化するVueコンポーネント。

型

- WjFlexPieDataLabel

WjFlexPie のwijmo.chart.PieDataLabel を表す Vueコンポーネント。

型

- WjFlexRadar

FlexRadar コントロールをカプセル化するVueコンポーネント。

型

- WjFlexRadarAxis

次のコンポーネントのいずれかに**FlexRadarAxis** を表すVueコンポーネント。 wijmo.vue2.WjFlexRadar 、またはwijmo.vue2.WjFinancialChart 。

型

- WjFlexRadarSeries

WjFlexRadar の**FlexRadarSeries** を表す Vueコンポーネント。

型

- WjFlexSheet

FlexSheet コントロールをカプセル化するVueコンポーネント。

型

WjFormat

グローバル化された書式設定を日付や数値に適用するVueフィルタ。

たとえば、次のコードは、**wj-format**フィルタを使用して、 現在のWijmoカルチャに基づいて数値を通貨値として書式設定し、 日付を短い日付として書式設定します。

```
<p>value: {{ theAmount | wj-format('c') }}</p>
<p>date: {{ theDate | wj-format('d') }}</p>
```

型

WjGroupPanel

GroupPanel コントロールをカプセル化するVueコンポーネント。

次の例は、Vueで**GroupPanel** と**FlexGrid** を インスタンス化して接続する方法を示します。

```
<wj-group-panel
  id="thePanel"
  placeholder="Drag columns here to create Groups">
</wj-group-panel>
<wj-flex-grid
  id="theGrid"
  :items-source="data">
</wj-flex-grid>
```

```
var app = new Vue({
  el: '#app',

  // connect group panel and grid
  mounted: function () {
    var panel = wijmo.Control.getControl(document.getElementById('thePanel'));
    var grid = wijmo.Control.getControl(document.getElementById('theGrid'));
    panel.grid = grid;
  }
});
```

型

WjInclude

特定のHTMLフラグメントをドキュメントに含めるVueコンポーネント。

wj-includeコンポーネントは、ドキュメントにロードおよび挿入する ファイルを指定する**src**属性を受け取ります。次に例を示します。

```
<wj-popup control="modalDialog" :modal="true" :hide-trigger="None">
  <wj-include src="includes/dialog.htm"></wj-include>
</wj-popup>
```

型

WjInputColor

InputColor コントロールをカプセル化するVueコンポーネント。

型

- WjInputDate

InputDate コントロールをカプセル化するVueコンポーネント。

型

- WjInputDateTime

InputDateTime コントロールをカプセル化するVueコンポーネント。

型

- WjInputMask

InputMask コントロールをカプセル化するVueコンポーネント。

型

- WjInputNumber

InputNumber コントロールをカプセル化するVueコンポーネント。

型

- WjInputTime

InputTime コントロールをカプセル化するVueコンポーネント。

型

- WjLinearGauge

LinearGauge コントロールをカプセル化するVueコンポーネント。

次の例は、Vueマークアップを使用して、 **LinearGauge** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-linear-gauge
    :min="0" :max="1000" :step="50" :is-read-only="false"
    format="c0" :thumb-size="20"
    :show-ranges="false"
    :value="sales"
    :value-changed="salesChanged">
    <wj-range wj-property="face" :thickness="0.5">
    </wj-range>
    <wj-range wj-property="pointer" :thickness="0.5">
    </wj-range>
    <wj-range :min="0" :max="333" color="red">
    </wj-range>
    <wj-range :min="333" :max="666" color="gold">
    </wj-range>
    <wj-range :min="666" :max="1000" color="green">
    </wj-range>
</wj-linear-gauge>
```

このコードは、**min**、**max**、**step**、**isReadOnly**の各プロパティを設定して、 ゲージの範囲を定義し、ユーザーがゲージの値を編集できるようにします。 次に、ゲージの**value**プロパティをコントローラーの**sales**変数に連結します。

次に、**format**, **thumbSize**、**showRanges**の各プロパティを設定して、 ゲージの外観を定義します。最後に、このマークアップは、 **face**範囲と**pointer**範囲の太さを設定し、ゲージの現在の値に応じて **value**範囲の色を制御するために、いくつかの範囲を設定します。

型

- WjListBox

ListBox コントロールをカプセル化するVueコンポーネント。

型

- WjMenu

Menu コントロールをカプセル化するVueコンポーネント。

型

- WjMultiAutoComplete

MultiAutoComplete コントロールをカプセル化するVueコンポーネント。

型

- WjMultiRow

MultiRow コントロールをカプセル化するVueコンポーネント。

型

- WjMultiSelect

MultiSelect コントロールをカプセル化するVueコンポーネント。

型

- WjPdfViewer

PdfViewer コントロールをカプセル化するVueコンポーネント。

型

- WjPivotChart

PivotChart コントロールをカプセル化するVueコンポーネント。

型

- WjPivotGrid

PivotGrid コントロールをカプセル化するVueコンポーネント。

型

- WjPivotPanel

PivotPanel コントロールをカプセル化するVueコンポーネント。

型

- WjPopup

Popup コントロールをカプセル化するVueコンポーネント。

型

- WjRadialGauge

RadialGauge コントロールをカプセル化するVueコンポーネント。

次の例は、Vueマークアップを使用して、 **RadialGauge** コントロールをインスタンス化して初期化する方法を示します。

```
<wj-radial-gauge
    :min="0" :max="1000" :step="50" :is-read-only="false"
    format="c0" :thumb-size="12" :show-text="Value"
    :show-ranges="false"
    :value="sales"
    :value-changed="salesChanged">
    <wj-range wj-property="face" :thickness="0.5">
    </wj-range>
    <wj-range wj-property="pointer" :thickness="0.5">
    </wj-range>
    <wj-range :min="0" :max="333" color="red">
    </wj-range>
    <wj-range :min="333" :max="666" color="gold">
    </wj-range>
    <wj-range :min="666" :max="1000" color="green">
    </wj-range>
</wj-radial-gauge>
```

このコードは、**min**、**max**、**step**、**isReadOnly**の各プロパティを設定して、 ゲージの範囲を定義し、ユーザーがゲージの値を編集できるようにします。 次に、ゲージの**value**プロパティをコントローラーの**sales**変数に連結します。

次に、**format**, **thumbSize**、**showRanges**の各プロパティを設定して、 ゲージの外観を定義します。最後に、このマークアップは、 **face**範囲と**pointer**範囲の太さを設定し、ゲージの現在の値に応じて **value**範囲の色を制御するために、いくつかの範囲を設定します。

型

- WjRange

次のコンポーネントのいずれかに**Range** を表すVueコンポーネント。 **WjLinearGauge** 、**WjBulletGraph**、 または**WjRadialGauge**.

型

- WjReportViewer

ReportViewer コントロールをカプセル化するVueコンポーネント。

型

- WjSheet

WjFlexSheet の**Sheet** を表す Vueコンポーネント。

型

- WjSlicer

Slicer コントロールをカプセル化するVueコンポーネント。

型

- WjSunburst

Sunburst コントロールをカプセル化するVueコンポーネント。

型

- WjTreeMap

TreeMap コントロールをカプセル化するVueコンポーネント。

型

- WjTreeView

TreeView コントロールをカプセル化するVueコンポーネント。

型