# DEVEXTREME

## 1 Basics of DevExtreme

### 1.1 What is DevExpress and DevExtreme?

* DevExpress is component vendor for variety of controls for different technologies.

### 1.2 Nuget Package installation

* Install-Package DevExtreme -Version 21.1.3

### 1.3 jQuery widgets

* Any devextreme UI component must be placed inside a container.
* dxButton is a jQuery plugin provided by devextreme to each UI component.
* To configure the UI component u have to pass **an object** to the plugin. The props of this object act as mirror to the props of the UI component.
* $(“#container”).dxButton() is different from $(“#conatiner”).dxButton(“instance”).
* At first place dx create a dx-button-content class div inside the conatiner and returns the conatiner jQuery element.
* At second, dx returns the button widget instance using which u can call dx’s method or access its properties.

### 1.4 Call methods of dx widgets

* There are several ways to invoke a dx UI component methods.

#### 1.4.1 Using the widget instance:

*myWidget.methodName();*

#### 1.4.2 Using the jQuery interface:

*$(“#myWidgetElement”).dxMyWidget(“methodName”);*

#### 1.4.3 Using DOM Element:

* First get raw widget instance,
* Then get the dx instance of that widget,
* And the it is as same as 1.4.1.

*var widgetElement = document.getElementById(“myWidgetElement”);*

*var widgetInstance = DevExpress.ui.getWidgetInstance(widgetElement);*

*widgetInstance.methodName();*

#### 1.4.3 Using global widget instance method

For some widget, there are global method available on ‘DevExpress.ui’, which can be used to interact with widgets.

E.g., DevExpress.ui.notify to show notification for some widget.

### 1.5 DevExpress.ui namespace

* DevExpress.ui is a namespace that contains global utility methods and properties related to UI components.
* Task performed by DevExpress.ui:
  1. Notification
  2. Widget Registraction
  3. Widget Lookup
  4. Widget Theme

### 1.6 Accessing custom handlers

*let buttonInstance = $("#container").dxButton("instance");*

*buttonInstance.\_options.\_optionManager.\_options.onClick*

*document.getElementById(“#container”).jQuery351039286975644576462.dxButton.\_options.\_optionManager.\_options.onClick*

* Another way to register an event in devextreme is using .on method. It allows you to subscribe to events at runtime and even to attach several handlers to a single event. E.g., myWidgetInstance.on(“click”, function(){  
   alert(“click1”);  
  });

myWidgetInstance.on(“click”, function(){  
 alert(“click2”);  
});

* To manually locate this handlers:

*document.getElementById(“#container”).jQuery351039286975644576462*.dxButton.\_eventsStrategy.\_events.click.\_list

#### Unsubscribing from an event handler

To unsubscribe from an event handler attached using .on() method. You have to invoke .off() method. (Both invokes \_eventStrategy methods only).

myWidgetInstance.off(“click”) => empty the \_list attached to click event in \_eventStrategy instance.

myWidgetInstance.off(“click”, handlerRef) => pops out the handlerRef from the \_list attached to click event in \_eventStrategy instance.

To unsubscribe from handler attached using .onEventName property, u have to do, myWidgetInstance.option(“eventName”, undefined);

## 2 Editors

### DevExtreme Components Options (Version 21.1)

#### Common Options

Data Binding

* dataSource: Specifies the data source for the component.
* value: Specifies the current value.
* valueExpr: Specifies which data field provides the component's value.
* displayExpr: Specifies which data field provides the display text.
* valueChangeEvent: Specifies the DOM events that trigger updates to the value.

Appearance

* elementAttr: Specifies the global attributes to be attached to the component's root element.
* height: Specifies the component's height.
* width: Specifies the component's width.
* visible: Specifies whether the component is visible.
* rtlEnabled: Switches the component to a right-to-left representation.

Behavior

* disabled: Specifies whether the component is disabled.
* readOnly: Specifies whether the component is read-only.
* tabIndex: Specifies the tab index of the component.

Events

* onValueChanged: A function that is executed after the component's value is changed.
* onInitialized: A function that is executed after the component is initialized.
* onDisposing: A function that is executed before the component is disposed of.

#### Check Box

##### Options

**accessKey**: Defines a shortcut key that, when pressed, focuses on the UI component.

**activeStateEnabled**: Determines if the UI component changes its state (appearance) when the user interacts with it.

**disabled**: Indicates whether the UI component should ignore user interactions.

**elementAttr**: Specifies global HTML attributes to be attached to the container element of the UI component.

**focusStateEnabled**: Indicates whether the UI component can gain focus through keyboard navigation.

**height**: Defines the height of the UI component.

**hint**: Sets the text for a tooltip that appears when the user hovers over the UI component.

**hoverStateEnabled**: Determines if the UI component changes its state when the user hovers over it.

**isValid**: Indicates or sets the validity of the editor's value.

**name**: Assigns a value to the name attribute of the underlying HTML element.

**onContentReady**: A callback function that is triggered when the UI component's content is ready or has changed.

**onDisposing**: A callback function that is triggered before the UI component is disposed of.

**onInitialized**: A callback function used to save the instance of the UI component, typically used in JavaScript frameworks.

**onOptionChanged**: A callback function that is triggered after a property of the UI component is changed.

**onValueChanged**: A callback function that is triggered after the UI component's value is changed.

**readOnly**: Indicates whether the editor is read-only, preventing any user modifications.

**rtlEnabled**: Switches the UI component to a right-to-left layout, often used for languages that are read from right to left.

**tabIndex**: Specifies the tab order of the element when navigating using the Tab key.

**text**: Defines the text displayed by the UI component.

**validationError**: Contains information about the first validation rule that failed.

**validationErrors**: An array containing all the validation rules that have failed.

**validationMessageMode**: Defines how the validation message is displayed when the editor's value does not meet validation rules.

**validationStatus**: Indicates or sets the current validation status of the editor.

**value**: Represents or sets the current value of the UI component.

**visible**: Determines whether the UI component is visible.

**width**: Defines the width of the UI component.

##### Methods

**beginUpdate():** Temporarily prevents the UI component from refreshing, which can be useful for making multiple changes to the component without causing it to refresh after each change.

**defaultOptions(rule):** Defines default configuration properties for the UI component that can be dependent on the device type.

**dispose():** Cleans up and releases all resources allocated by the UI component instance, ensuring no memory leaks.

**element():** Returns the root DOM element of the UI component, allowing access to the component's container.

**endUpdate():** Re-enables the UI component's ability to refresh after being disabled by beginUpdate(), causing it to refresh and apply all pending updates.

**focus():** Sets the focus to the UI component, making it the active element on the page.

**getInstance(element):** Retrieves the instance of a UI component associated with a specific DOM element.

**instance():** Returns the current instance of the UI component, which can be used to call other methods or access properties of the component.

**off(eventName):** Removes all event handlers associated with a specific event from the UI component.

**off(eventName, eventHandler):** Removes a specific event handler from a specific event.

**on(eventName, eventHandler):** Adds an event handler for a specific event to the UI component.

**on(events):** Adds multiple event handlers for multiple events to the UI component using an object where keys are event names and values are event handlers.

**option():** Retrieves all the properties and their current values of the UI component.

**option(optionName):** Retrieves the value of a specific property of the UI component.

**option(optionName, optionValue):** Sets a new value for a specific property of the UI component.

**option(options):** Updates multiple properties of the UI component with new values, provided as an object.

**registerKeyHandler(key, handler):** Registers a handler function to be executed when the user presses a specific key while the UI component is focused.

**repaint():** Redraws the UI component without reloading its data, useful for updating the component's appearance or layout after making changes.

**reset():** Resets the value property of the UI component to its default value.

**resetOption(optionName):** Resets a specific property of the UI component to its default value.

##### Events

**contentReady:** This event is triggered when the UI component's content has been fully loaded and is ready for interaction. It can be used to execute custom logic after the content is prepared.

**disposing:** This event is fired just before the UI component is disposed of. It's useful for performing any necessary cleanup or final actions before the component is removed from the DOM and its resources are released.

**initialized:** This event occurs once, immediately after the UI component has been initialized. It's typically used to perform actions that need to happen as soon as the component is ready.

**optionChanged:** This event is triggered whenever a property of the UI component is changed. It allows you to respond dynamically to changes in the component's configuration.

**valueChanged:** This event occurs after the value of the UI component has been modified. It's useful for executing code in response to changes in the component's value, such as validating input or updating related data.

#### Date Box

##### Options

**Calendar-Specific Options:**

**calendarOptions:** Configures the calendar's value picker. Applies only if the pickerType is "calendar".

**dateOutOfRangeMessage:** Specifies the message displayed if the specified date is later than the max value or earlier than the min value.

**disabledDates:** Specifies dates that users cannot select. Applies only if pickerType is "calendar".

**displayFormat**: Specifies the date display format. Ignored if the pickerType property is "native".

**invalidDateMessage:** Specifies the message displayed if the typed value is not a valid date or time.

**max:** The last date that can be selected within the UI component.

**min:** The minimum date that can be selected within the UI component.

**pickerType:** Specifies the type of the date/time picker.

**showAnalogClock:** Specifies whether to show the analog clock in the value picker. Applies only if type is "datetime" and pickerType is "calendar".

**type**: A format used to display date/time information.

**Input-Specific Options:**

**acceptCustomValue:** Specifies whether or not the UI component allows an end-user to enter a custom value.

**accessKey**: Specifies the shortcut key that sets focus on the UI component.

**activeStateEnabled:** Specifies whether the UI component changes its state as a result of user interaction.

**applyButtonText:** The text displayed on the Apply button.

**applyValueMode:** Specifies the way an end-user applies the selected value.

**buttons:** Allows you to add custom buttons to the input text field.

**cancelButtonText:** The text displayed on the Cancel button.

**dateSerializationFormat:** Specifies the date-time value serialization format. Use it only if you do not specify the value at design time.

**deferRendering:** Specifies whether to render the drop-down field's content when it is displayed. If false, the content is rendered immediately.

**disabled:** Specifies whether the UI component responds to user interaction.

**dropDownButtonComponent:** An alias for the dropDownButtonTemplate property specified in React. Accepts a custom component. Refer to Using a Custom Component for more information.

**dropDownButtonRender:** An alias for the dropDownButtonTemplate property specified in React. Accepts a rendering function. Refer to Using a Rendering Function for more information.

**dropDownButtonTemplate:** Specifies a custom template for the drop-down button.

**dropDownOptions**: Configures the drop-down field which holds the content.

**elementAttr**: Specifies the global attributes to be attached to the UI component's container element.

**focusStateEnabled**: Specifies whether the UI component can be focused using keyboard navigation.

**height**: Specifies the UI component's height.

**hint**: Specifies text for a hint that appears when a user pauses on the UI component.

**hoverStateEnabled**: Specifies whether the UI component changes its state when a user pauses on it.

**inputAttr**: Specifies the attributes to be passed on to the underlying HTML element.

**interval**: Specifies the interval between neighboring values in the popup list in minutes.

**isValid**: Specifies or indicates whether the editor's value is valid.

**maxLength**: Specifies the maximum number of characters you can enter into the textbox.

**name**: The value to be assigned to the name attribute of the underlying HTML element.

**opened**: Specifies whether or not the drop-down editor is displayed.

**openOnFieldClick**: Specifies whether a user can open the drop-down list by clicking a text field.

**placeholder**: Specifies a placeholder for the input field.

**readOnly**: Specifies whether the editor is read-only.

**rtlEnabled**: Switches the UI component to a right-to-left representation.

**showClearButton**: Specifies whether to display the Clear button in the UI component.

**showDropDownButton**: Specifies whether the drop-down button is visible.

**spellcheck**: Specifies whether or not the UI component checks the inner text for spelling mistakes.

**stylingMode**: Specifies how the UI component's text field is styled.

**tabIndex**: Specifies the number of the element when the Tab key is used for navigating.

**text**: The read-only property that holds the text displayed by the UI component input element.

**useMaskBehavior**: Specifies whether to control user input using a mask created based on the displayFormat.

**validationError**: Information on the broken validation rule. Contains the first item from the validationErrors array.

**validationErrors**: An array of the validation rules that failed.

**validationMessageMode**: Specifies how the message about the validation rules that are not satisfied by this editor's value is displayed.

**validationStatus**: Indicates or specifies the current validation status.

**value**: An object or a value specifying the date and time currently selected using the date box.

**valueChangeEvent**: Specifies the DOM events after which the UI component's value should be updated.

**visible**: Specifies whether the UI component is visible.

**width**: Specifies the UI component's width.

**Event-Specific Options**:

**onChange**: A function that is executed when the UI component loses focus after the text field's content was changed using the keyboard.

**onClosed**: A function that is executed once the drop-down editor is closed.

**onContentReady**: A function that is executed when the UI component's content is ready and each time the content is changed.

**onCopy**: A function that is executed when the UI component's input has been copied.

**onCut**: A function that is executed when the UI component's input has been cut.

**onDisposing**: A function that is executed before the UI component is disposed of.

**onEnterKey**: A function that is executed when the Enter key has been pressed while the UI component is focused.

**onFocusIn**: A function that is executed when the UI component gets focus.

**onFocusOut**: A function that is executed when the UI component loses focus.

**onInitialized**: A function used in JavaScript frameworks to save the UI component instance.

**onInput**: A function that is executed each time the UI component's input is changed while the UI component is focused.

**onKeyDown**: A function that is executed when a user is pressing a key on the keyboard.

**onKeyUp**: A function that is executed when a user releases a key on the keyboard.

**onOpened**: A function that is executed once the drop-down editor is opened.

**onOptionChanged**: A function that is executed after a UI component property is changed.

**onPaste**: A function that is executed when the UI component's input has been pasted.

**onValueChanged**: A function that is executed after the UI component's value is changed.

##### Methods

**beginUpdate**(): Prevents the UI component from refreshing until the endUpdate() method is called.

**blur**(): Removes focus from the input element.

**close**(): Closes the drop-down editor.

**content**(): Gets the popup window's content.

**defaultOptions**(rule): Specifies the device-dependent default configuration properties for this component.

**dispose**(): Disposes of all the resources allocated to the DateBox instance.

**element**(): Gets the root UI component element.

**endUpdate**(): Refreshes the UI component after a call of the beginUpdate() method.

**field**(): Gets the UI component's <input> element.

**focus**(): Sets focus to the input element representing the UI component.

**getButton**(name): Gets an instance of a custom action button.

**getInstance**(element): Gets the instance of a UI component found using its DOM node.

**instance**(): Gets the UI component's instance. Use it to access other methods of the UI component.

**off**(eventName): Detaches all event handlers from a single event.

**off**(eventName, eventHandler): Detaches a particular event handler from a single event.

**on**(eventName, eventHandler): Subscribes to an event.

**on**(events): Subscribes to events.

**open**(): Opens the drop-down editor.

**option**(): Gets all UI component properties.

**option**(optionName): Gets the value of a single property.

**option**(optionName, optionValue): Updates the value of a single property.

**option**(options): Updates the values of several properties.

**registerKeyHandler**(key, handler): Registers a handler to be executed when a user presses a specific key.

**repaint**(): Repaints the UI component without reloading data. Call it to update the UI component's markup.

**reset**(): Resets the value property to the default value.

**resetOption**(optionName): Resets a property to its default value.

### Events

**onValueChanged**: Occurs when the widget's value changes.

**onOpened**: Occurs when the drop-down calendar is opened.

**onClosed**: Occurs when the drop-down calendar is closed.

#### Drop Down Box

##### Options

**acceptCustomValue**: Specifies whether the widget allows an end-user to enter a custom value.

**accessKey**: Specifies the shortcut key that sets focus on the widget.

**activeStateEnabled**: Specifies whether the widget changes its state when interacting with a user.

**contentTemplate**: Specifies a custom template for the drop-down content.

**dataSource**: Specifies the data source for the widget.

**deferRendering**: Specifies whether to render the drop-down content when it is displayed.

**displayExpr**: Specifies the data field whose values should be displayed.

**dropDownOptions**: Specifies options for the drop-down list.

**fieldTemplate**: Specifies a custom template for the text field.

**focusStateEnabled**: Specifies whether the widget can be focused.

**height**: Specifies the widget's height.

**hint**: Specifies the text displayed when the mouse pointer is over the widget.

**items**: An array of items to be displayed by the widget.

**opened**: Specifies whether the drop-down is opened.

**placeholder**: The text displayed by the widget when the widget is empty.

**readOnly**: Specifies whether the widget is read-only.

**searchEnabled**: Specifies whether or not the widget supports searching.

**showClearButton**: Specifies whether the widget displays a button that clears the value.

**tabIndex**: Specifies the number of the element when the Tab key is used for navigating.

##### Methods

**blur**(): Removes focus from the widget.

**close**(): Closes the drop-down.

**focus**(): Sets focus to the widget.

**open**(): Opens the drop-down.

**repaint**(): Redraws the widget.

##### Events

**onValueChanged**: Occurs when the widget's value changes.

**onOpened**: Occurs when the drop-down is opened.

**onClosed**: Occurs when the drop-down is closed.

#### **Number** Box

##### Options

**accessKey**: Specifies the shortcut key that sets focus on the widget.

**activeStateEnabled**: Specifies whether the widget changes its state when interacting with a user.

**buttons**: Specifies options for the spin buttons.

**format**: Specifies the format string that is used to format the value displayed by the widget.

**invalidValueMessage**: Specifies the message displayed in the tooltip if the widget's value is not a valid number.

**max**: The maximum value that can be entered into the widget.

**min**: The minimum value that can be entered into the widget.

**mode**: Specifies how the widget's value can be changed using the spin buttons.

**placeholder**: The text displayed by the widget when the widget is empty.

**readOnly**: Specifies whether the widget is read-only.

**showSpinButtons**: Specifies whether the spin buttons are visible.

**step**: Specifies the step by which the widget's value changes when using spin buttons.

**value**: Specifies the value of the widget.

##### Methods

**blur**(): Removes focus from the widget.

**focus**(): Sets focus to the widget.

##### Events

**onValueChanged**: Occurs when the widget's value changes.

#### Select Box

##### Options

**accessKey**: Specifies the shortcut key that sets focus on the widget.

**activeStateEnabled**: Specifies whether the widget changes its state when interacting with a user.

**dataSource**: Specifies the data source for the widget.

**displayExpr**: Specifies which data field provides display values for items.

**disabled**: Specifies whether the widget is disabled.

**dropDownOptions**: Specifies options for the drop-down list.

**focusStateEnabled**: Specifies whether the widget can be focused.

**grouped**: Specifies whether or not the drop-down items should be grouped.

**groupTemplate**: Specifies a custom template for group headers.

**height**: Specifies the widget's height.

**hint**: Specifies the text displayed when the mouse pointer is over the widget.

**itemTemplate**: Specifies a custom template for items.

**items**: An array of items to be displayed by the widget.

### Text Area

**Options:**

**accessKey**: Specifies the shortcut key that sets focus on the UI component. This is a single character that, when pressed along with a modifier key (like Alt or Ctrl), allows the user to quickly access the UI component.

**activeStateEnabled**: Specifies whether the UI component changes its state as a result of user interaction. When enabled, the UI component visually responds to user actions like hovering or clicking.

**autoResizeEnabled**: A Boolean value specifying whether or not the auto resizing mode is enabled. When enabled, the UI component automatically adjusts its size based on its content.

**disabled**: Specifies whether the UI component responds to user interaction. When disabled, the UI component is non-interactive and may display differently to indicate its disabled state.

**elementAttr**: Specifies the global attributes to be attached to the UI component's container element. These attributes can be used to customize the behavior or appearance of the container element.

**focusStateEnabled**: Specifies whether the UI component can be focused using keyboard navigation. When enabled, the UI component can receive focus using the Tab key.

**height**: Specifies the UI component's height. This can be set to a specific value or a relative value like "auto" to allow the component to determine its own height based on its content.

**hint**: Specifies text for a hint that appears when a user pauses on the UI component. The hint provides additional information about the UI component's purpose or functionality.

**hoverStateEnabled**: Specifies whether the UI component changes its state when a user pauses on it. When enabled, the UI component may visually change to indicate its hover state.

**inputAttr**: Specifies the attributes to be passed on to the underlying HTML element. These attributes can be used to further customize the behavior or appearance of the underlying input element.

**isValid**: Specifies or indicates whether the editor's value is valid. This property is typically used in validation scenarios to check if the editor's value meets certain criteria.

**maxHeight**: Specifies the maximum height of the UI component. This can be used to limit the size of the component when it contains a large amount of content.

**maxLength**: Specifies the maximum number of characters you can enter into the textbox. This property is commonly used with text input components to limit the length of user input.

**minHeight**: Specifies the minimum height of the UI component. This can be used to ensure that the component is always at least a certain size, even when it contains minimal content.

**name**: The value to be assigned to the name attribute of the underlying HTML element. This can be used for form submission purposes to identify the UI component in the form data.

**onChange**: A function that is executed when the UI component loses focus after the text field's content was changed using the keyboard. This allows you to perform actions when the value of the UI component changes.

**onContentReady**: A function that is executed when the UI component's content is ready and each time the content is changed. This can be used to perform initialization or other actions when the component is ready to be used.

**onCopy**, **onCut**, **onPaste**: Functions that are executed when the UI component's input has been copied, cut, or pasted, respectively. These can be used to perform custom actions when these clipboard operations are performed on the UI component.

**onDisposing**: A function that is executed before the UI component is disposed of. This can be used to perform cleanup or other actions before the component is removed from the DOM.

**onEnterKey**: A function that is executed when the Enter key has been pressed while the UI component is focused. This allows you to perform actions when the Enter key is used to submit or confirm input.

**onFocusIn**, onFocusOut: Functions that are executed when the UI component gets focus or loses focus, respectively. These can be used to perform actions when the UI component gains or loses focus.

**onInitialized**: A function used in JavaScript frameworks to save the UI component instance. This can be used to access the UI component instance from other parts of your application.

**onInput**: A function that is executed each time the UI component's input is changed while the UI component is focused. This allows you to perform actions in response to user input.

**onKeyDown**, onKeyUp: Functions that are executed when a user is pressing a key on the keyboard or releases a key on the keyboard, respectively. These can be used to perform actions in response to keyboard input.

**onOptionChanged**: A function that is executed after a UI component property is changed. This can be used to perform actions when a specific property of the UI component changes.

**onValueChanged**: A function that is executed after the UI component's value is changed. This allows you to perform actions when the value of the UI component changes.

**placeholder**: The text displayed by the UI component when the UI component value is empty. This can be used to provide a hint or example of the expected input.

**readOnly**: Specifies whether the editor is read-only. When set to true, the UI component cannot be edited by the user.

**rtlEnabled**: Switches the UI component to a right-to-left representation. When enabled, the UI component's layout and behavior are adjusted for right-to-left languages.

**spellcheck**: Specifies whether or not the UI component checks the inner text for spelling mistakes. When enabled, the UI component may underline misspelled words or provide suggestions for corrections.

**stylingMode**: Specifies how the UI component's text field is styled. This can be set to "filled" to use a filled style or "outlined" to use an outlined style, for example.

**tabIndex**: Specifies the number of the element when the Tab key is used for navigating. This can be used to control the order in which elements receive focus when the Tab key is pressed.

**text**: The read-only property that holds the text displayed by the UI component input element. This property can be used to retrieve the current text value of the UI component.

**validationError**: Information on the broken validation rule. Contains the first item from the validationErrors array. This can be used to provide feedback to the user when a validation rule is not satisfied.

**validationErrors**: An array of the validation rules that failed. This can be used to provide detailed information about which validation rules were not satisfied by the UI component's value.

**validationMessageMode**: Specifies how the message about the validation rules that are not satisfied by this editor's value is displayed. This can be set to "auto" to display the message automatically or "manual" to display it manually.

**validationStatus**: Indicates or specifies the current validation status. This can be used to check if the UI component's value is valid or to set the validation status manually.

**value**: Specifies a value the UI component displays. This can be used to set or retrieve the current value of the UI component.

**valueChangeEvent**: Specifies the DOM events after which the UI component's value should be updated. This can be used to customize when the UI component's value is updated in response to user input.

**visible**: Specifies whether the UI component is visible. When set to false, the UI component is hidden from view.

**width**: Specifies the UI component's width. This can be set to a specific value or a relative value like "auto" to allow the component to determine its own width based on its content.

### Text Box

**Optionxs:**

**mask**: The editor mask that specifies the custom format of the entered string.

**maskChar**: Specifies a mask placeholder. A single character is recommended.

**maskInvalidMessage**: A message displayed when the entered text does not match the specified pattern.

**maskRules**: Specifies custom mask rules.

**maxLength**: Specifies the maximum number of characters you can enter into the textbox

**mode**: The "mode" attribute value of the actual HTML input element representing the text box.

**showClearButton**: Specifies whether to display the Clear button in the UI component.

**showMaskMode**: Specifies when the UI component shows the mask. Applies only if useMaskedValue is true.

**spellcheck**: Specifies whether or not the UI component checks the inner text for spelling mistakes.

**stylingMode**: Specifies how the UI component's text field is styled.

**tabIndex**: Specifies the number of the element when the Tab key is used for navigating.

**text**: The read-only property that holds the text displayed by the UI component input element.

**useMaskedValue**: Specifies whether the value should contain mask characters or not.

Button

Options

**accessKey**: Specifies the shortcut key that sets focus on the UI component. This allows users to quickly navigate to the component using a keyboard shortcut.

**activeStateEnabled**: Specifies whether the UI component changes its state as a result of user interaction. When enabled, the component may visually change to indicate its active state, such as when it is clicked or hovered over.

**disabled**: Specifies whether the UI component responds to user interaction. When disabled, the component cannot be interacted with by the user.

**elementAttr**: Specifies the global attributes to be attached to the UI component's container element. These attributes can be used to customize the behavior or appearance of the container element.

**focusStateEnabled**: Specifies whether the UI component can be focused using keyboard navigation. When enabled, the component can receive focus using the Tab key.

**height**: Specifies the UI component's height. This can be set to a specific value or a relative value like "auto" to allow the component to determine its own height based on its content.

**hint**: Specifies text for a hint that appears when a user pauses on the UI component. The hint provides additional information about the UI component's purpose or functionality.

**hoverStateEnabled**: Specifies whether the UI component changes its state when a user pauses on it. When enabled, the component may visually change to indicate its hover state.

**icon**: Specifies the icon to be displayed on the button. This can be used to provide a visual indication of the button's purpose or function.

**onClick**: A function that is executed when the Button is clicked or tapped. This allows you to perform actions when the button is clicked.

**onContentReady**: A function that is executed when the UI component's content is ready and each time the content is changed. This can be used to perform initialization or other actions when the component is ready to be used.

**onDisposing**: A function that is executed before the UI component is disposed of. This can be used to perform cleanup or other actions before the component is removed from the DOM.

**onInitialized**: A function used in JavaScript frameworks to save the UI component instance. This can be used to access the UI component instance from other parts of your application.

**onOptionChanged**: A function that is executed after a UI component property is changed. This can be used to perform actions when a specific property of the UI component changes.

**rtlEnabled**: Switches the UI component to a right-to-left representation. When enabled, the component's layout and behavior are adjusted for right-to-left languages.

**stylingMode**: Specifies how the button is styled. This can be set to "text", "outlined", or "contained" to define the button's style.

**tabIndex**: Specifies the number of the element when the Tab key is used for navigating. This can be used to control the order in which elements receive focus when the Tab key is pressed.

**template**: Specifies a custom template for the Button UI component. This allows you to customize the appearance and content of the button.

**text**: The text displayed on the button. This is the visible text that users see on the button.

**type**: Specifies the button type. This can be set to "default", "normal", "back", "danger", or "success" to define the button's type.

**useSubmitBehavior**: Specifies whether the button submits an HTML form. When enabled, the button behaves like a submit button in an HTML form.

**validationGroup**: Specifies the name of the validation group to be accessed in the click event handler. This can be used to perform validation specific to a certain group of elements.

**visible**: Specifies whether the UI component is visible. When set to false, the UI component is hidden from view.

**width**: Specifies the UI component's width. This can be set to a specific value or a relative value like "auto" to allow the component to determine its own width based on its content.

### Methods

**beginUpdate()**

**defaultOptions(rule)**

**dispose()**

**element()**

**endUpdate()**

**focus()**

**getInstance(element)**

**instance()**

**off(eventName)**

**off(eventName, eventHandler)**

**on(eventName, eventHandler)**

**on(events)**

**option()**

**option(optionName)**

**option(optionName, optionValue)**

**option(options)**

**registerKeyHandler(key, handler)**

**repaint()**

**resetOption(optionName)**

Events  
**click**

**contentReady**

**disposing**

**initialized**

**optionChanged**

## File Uploader

### Options

**abortUpload**: A function that cancels the file upload process.

**accept**: Specifies the file types accepted by the file uploader.

**accessKey**

**activeStateEnabled**

**allowCanceling**: Specifies if an end user can remove a file from the selection and interrupt uploading.

**allowedFileExtensions**: Restricts file extensions that can be uploaded to the server.

**chunkSize**: Specifies the chunk size in bytes for chunked file uploads.

**dialogTrigger**: Specifies the HTML element that invokes the file upload dialog.

**disabled**

**dropZone**: Specifies the HTML element in which users can drag and drop files for upload.

**elementAttr**

**focusStateEnabled**

**height**

**hint**

**hoverStateEnabled**

**icon**: Specifies the icon to be displayed on the button.

**onClick**: A function that is executed when the file uploader button is clicked or tapped.

**onContentReady**

**onDisposing**

**onInitialized**

**onOptionChanged**

**rtlEnabled**

**stylingMode**

**tabIndex**

**template**: Specifies a custom template for the file uploader.

**text**

**type**: Specifies the file uploader type.

**useSubmitBehavior**: Specifies whether the file uploader submits an HTML form.

**validationGroup**: Specifies the name of the validation group to be accessed in the click event handler.

**visible**: Specifies whether the file uploader is visible.

**Width**

### Methods

**abortUpload**(): Cancels the file upload.

**abortUpload**(file): Cancels the file upload.

**abortUpload**(fileIndex): Cancels the file upload.

**getInstance**(element): Gets the instance of a UI component found using its DOM node.

**removeFile**(file): Removes a file.

**removeFile**(fileIndex): Removes a file with the specified index.

**upload**(): Uploads all the selected files.

**upload**(file): Uploads the specified file.

**upload**(fileIndex): Uploads a file with the specified index.

**beginUpdate**()

**defaultOptions**(rule)

**dispose**()

**element**()

**endUpdate**()

**focus()**

**instance**()

**off**(eventName)

**off**(eventName, eventHandler)

**on**(eventName, eventHandler)

**on**(events)

**option**()

**option(optionName)**

**option**(optionName, optionValue)

**option**(options)

**registerKeyHandler**(key, handler)

**repaint**()

**reset**()

**resetOption**(optionName)

## Validator

### Options

**adapter**: An object that specifies what and when to validate, and how to apply the validation result.

**elementAttr**

**height**

**name**

**onDisposing**

**onInitialized**

**onOptionChanged**

**onValidated**: A function that is executed after a value is validated.

**validationGroup**: Specifies the validation group the editor will be related to.

**validationRules**: An array of validation rules to be checked for the editor with which the dxValidator object is associated.

**width**: Specifies the UI component's width.

#### Methods

**dispose**()

**element**()

**focus**()

**getInstance**(element)

**instance**()

**off**(eventName)

**off**(eventName, eventHandler)

**on**(eventName, eventHandler)

**on**(events)

**option**()

**option**(optionName)

**option**(optionName, optionValue)

**option**(options)

**reset**(): Resets the value and validation result of the editor associated with the current Validator object.

**resetOption**(optionName)

**validate**(): Validates the value of the editor that is controlled by the current Validator object against the list of the specified validation rules.

## Radio Group

### Options

**accessKey**: Specifies the shortcut key that sets focus on the UI component.

**activeStateEnabled**: Specifies whether the UI component changes its state as a result of user interaction.

**dataSource**: Binds the UI component to data.

**disabled: Specifies whether the UI component responds to user interaction.**

**displayExpr**: Specifies the data field whose values should be displayed.

**elementAttr**: Specifies the global attributes to be attached to the UI component's container element.

**focusStateEnabled**: Specifies whether the UI component can be focused using keyboard navigation.

**height**: Specifies the UI component's height.

**hint**: Specifies text for a hint that appears when a user pauses on the UI component.

**hoverStateEnabled**: Specifies whether the UI component changes its state when a user pauses on it.

**isValid**: Specifies or indicates whether the editor's value is valid.

**items**: An array of items displayed by the UI component.

**itemTemplate**: Specifies a custom template for items.

**layout**: Specifies the radio group layout.

**name**: The value to be assigned to the name attribute of the underlying HTML element.

**onContentReady**: A function that is executed when the UI component's content is ready and each time the content is changed.

**onDisposing**: A function that is executed before the UI component is disposed of.

**onInitialized**: A function used in JavaScript frameworks to save the UI component instance.

**onOptionChanged**: A function that is executed after a UI component property is changed.

**onValueChanged**: A function that is executed after the UI component's value is changed.

**readOnly**: Specifies whether the editor is read-only.

**rtlEnabled**: Switches the UI component to a right-to-left representation.

**tabIndex**: Specifies the number of the element when the Tab key is used for navigating.

**validationError**: Information on the broken validation rule. Contains the first item from the validationErrors array.

**validationErrors**: An array of the validation rules that failed.

**validationMessageMode**: Specifies how the message about the validation rules that are not satisfied by this editor's value is displayed.

**validationStatus**: Indicates or specifies the current validation status.

**value**: Specifies the UI component's value.

**valueExpr**: Specifies which data field provides unique values to the UI component's value.

**visible**: Specifies whether the UI component is visible.

**width**: Specifies the UI component's width.

### Methods

**beginUpdate**()

**defaultOptions**(rule)

**dispose()**

**element**)

**endUpdate**()

**focus**()

**getDataSource**()

**getInstance**(element)

**instance**()

**off**(eventName)

**off**(eventName, eventHandler)

**on**(eventName, eventHandler)

**on**(events)

**option**()

**option**(optionName)

**option**(optionName, optionValue)

**option(options)**

**registerKeyHandler**(key, handler)

**repaint**()

**reset**()

**resetOption**(optionName)