**Create and configure a widget:**

Devextreme UI component must be placed in a div container. Add the widget by chaining them on the selector with prefix as ‘dx’.

**Get and Set properties:**

We can get single property, all properties, set single property or get all properties. On the instance of the widget, one can retrieve or set the property or multiple properties by passing them in an object.

**Call methods:**

One can call various methods from the widget instance such as reset, focus, etc.

**Handle Events:**

There are various events that can be attached to a widget by subscribing them using ‘on’ method. To unsubscribe an event, pass the event or function name as 2nd parameter in the ‘off’ method.

There are various events such as :onCellClick, onKeyDown, onFocus, etc.

**Destroy a widget:**

A widget can be destroyed by invoking the ‘dispose()’ method on the widget instance and further use .remove() method for removing the corresponding ‘div’ tag of that widget from the DOM.

**option in DevExtreme?**

In DevExtreme, option is a method used to **get or set** configuration options dynamically for UI components **after initialization**. This allows you to update the component's properties without needing to recreate it.

Options: can be modified. Option method for initialization.

Properties: read-only. Use widget instance methods to get the property details.

var dataGridInstance = $("#dataGridContainer").dxDataGrid("instance");

        var dataSource = dataGridInstance.option("dataSource");

        var editMode = dataGridInstance.option("editing.mode");

        // ---------- or ----------

        var dataSource = $("#dataGridContainer").dxDataGrid("option", "dataSource");

        var editMode = $("#dataGridContainer").dxDataGrid("option", "editing.mode");

A screenshot of a black screen

Description automatically generated

**Which One Should You Use?**

1. **Use Version 1 (instance) when:**
   * You are **accessing multiple options** from the DataGrid.
   * You might need to **call other methods** on the DataGrid instance later.
   * You want better **performance** for repeated interactions.
2. **Use Version 2 (Direct .option) when:**
   * You only need to retrieve **one or two options**.
   * You prefer a **shorter** and simpler syntax.
   * Performance is not a major concern.

Widgets:

1. **Checkbox:**

**Options**

* text
* value
* enableThreeStateBehavior
* disabled
* hint
* iconSize
* activeStateEnabled
* focusStateEnabled
* hoverStateEnabled
* isValid
* name
* readOnly
* rtlEnabled
* tabIndex
* validationStatus
* validationErrors
* visible
* width
* accessKey
* elementAttr

**Events**

* onValueChanged
* onOptionChanged
* onInitialized
* onContentReady
* onDisposing

**Methods**

* option()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* reset()
* resetOption()
* element()
* focus()
* registerKeyHandler()
* off()
* on()

1. **Datebox:**

**Options**

* value
* min
* max
* displayFormat
* type
* opened
* readOnly
* disabled
* showClearButton
* placeholder
* useMaskBehavior
* showAnalogClock
* show24HourFormat
* focusStateEnabled
* activeStateEnabled
* tabIndex
* validationStatus
* validationErrors
* hint
* elementAttr
* rtlEnabled
* width
* height
* name
* displayExpr
* valueExpr

**Events**

* onValueChanged
* onOpened
* onClosed
* onFocusIn
* onFocusOut
* onInitialized
* onOptionChanged
* onContentReady
* onDisposing

**Methods**

* option()
* reset()
* resetOption()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* element()
* focus()
* selectText()
* clear()
* show()
* hide()

1. **Dropdownbox:**

**Options**

* value
* valueExpr
* displayExpr
* dataSource
* placeholder
* searchEnabled
* showClearButton
* inputAttr
* opened
* readOnly
* disabled
* clearButtonText
* useSearch
* grouped
* groupTemplate
* contentTemplate
* height
* width
* rtlEnabled
* tabIndex
* validationStatus
* validationErrors
* showPopupButton
* popupHeight
* popupWidth
* searchMode
* minSearchLength
* showTitle

**Events**

* onValueChanged
* onOpened
* onClosed
* onContentReady
* onDisposing
* onOptionChanged
* onInitialized
* onSelectionChanged
* onSearch

**Methods**

* option()
* reset()
* resetOption()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* element()
* focus()
* selectText()
* clear()
* show()
* hide()
* open()
* close()

1. **Numberbox:**

**Options**

* value
* valueExpr
* displayExpr
* min
* max
* step
* placeholder
* showSpinButtons
* format
* mode
* useLargeSpinButtons
* showClearButton
* inputAttr
* disabled
* readOnly
* rtlEnabled
* tabIndex
* width
* height
* validationStatus
* validationErrors

**Events**

* onValueChanged
* onInput
* onFocusIn
* onFocusOut
* onKeyDown
* onKeyUp
* onContentReady
* onDisposing
* onInitialized
* onOptionChanged

**Methods**

* option()
* reset()
* resetOption()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* element()
* focus()
* clear()
* show()
* hide()

1. **Selectbox:**

**Options**

* value
* dataSource
* displayExpr
* valueExpr
* placeholder
* searchEnabled
* searchMode
* searchTimeout
* minSearchLength
* itemTemplate
* grouped
* groupTemplate
* showClearButton
* readOnly
* disabled
* showDataBeforeSearch
* inputAttr
* clearButtonText
* showSelectionControls
* dropDownButtonTemplate
* width
* height
* validationStatus
* validationErrors
* hint
* tabIndex
* itemHoldTimeout
* deferRendering
* openOnFieldClick
* opened
* dropDownOptions
* popupHeight
* popupWidth
* useItemTextAsTitle

**Events**

* onValueChanged
* onOpened
* onClosed
* onItemClick
* onItemContextMenu
* onItemHold
* onContentReady
* onDisposing
* onInitialized
* onOptionChanged

**Methods**

* option()
* reset()
* resetOption()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* element()
* focus()
* clear()
* show()
* hide()
* open()
* close()

**6) Textarea:**

**Options**

* accessKey
* activeStateEnabled
* autoResizeEnabled
* disabled
* elementAttr
* focusStateEnabled
* height
* hint
* hoverStateEnabled
* inputAttr
* isValid
* maxHeight
* maxLength
* minHeight
* name
* placeholder
* readOnly
* rtlEnabled
* spellCheck
* stylingMode
* tabIndex
* text
* validationError
* validationErrors
* validationMessageMode
* validationStatus
* value
* valueChangeEvent
* visible
* width

**Methods**

* blur()
* focus()
* reset()
* select()
* updateDimensions()

**Events**

* onChange
* onContentReady
* onCopy
* onCut
* onDisposing
* onEnterKey
* onFocusIn
* onFocusOut
* onInitialized
* onInput
* onKeyDown
* onKeyPress
* onKeyUp
* onOptionChanged
* onPaste
* onValueChanged

**7) Textbox:**

**Options**

* accessKey
* activeStateEnabled
* buttons
* disabled
* elementAttr
* focusStateEnabled
* height
* hint
* hoverStateEnabled
* inputAttr
* isValid
* label
* labelMode
* mask
* maskChar
* maskInvalidMessage
* maskRules
* maxLength
* mode
* name
* placeholder
* readOnly
* rtlEnabled
* showClearButton
* showMaskMode
* spellCheck
* stylingMode
* tabIndex
* text
* useMaskedValue
* validationError
* validationErrors
* validationMessageMode
* validationStatus
* value
* valueChangeEvent
* visible
* width

**Methods**

* blur()
* focus()
* getButton(name)
* reset()
* select()
* updateDimensions()

**Events**

* onChange
* onContentReady
* onCopy
* onCut
* onDisposing
* onEnterKey
* onFocusIn
* onFocusOut
* onInitialized
* onInput
* onKeyDown
* onKeyPress
* onKeyUp
* onOptionChanged
* onPaste
* onValueChanged

--

Validation Rules:

1. Async rules:

A custom validation rule that is checked asynchronously. Use async rules for server-side validation. (Eg: This username is already taken.)

1. Compare rules:

A validation rule that demands that a validated editor has a value that is equal to a specified expression.

1. Custom rules: Pass custom logic in validationCallback.
2. Email rule: Built-in for email verification.
3. Numeric rule: Built-in numeric check verification.
4. Pattern rule: Pass custom reg-ex for verification.
5. Range rule: Specifies range for integral input.
6. Required rule: Specifies that the following widget value is required.
7. Stringlength rule: Same as range but for string length