**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: these 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:**

1. **text**: The text displayed next to the checkbox.
2. **value**: The current value of the checkbox. If checked, the value is true; if unchecked, the value is false.
3. **enableThreeStateBehavior**: Enables a third state for the checkbox (indeterminate).
4. **disabled**: Disables the checkbox, making it unclickable.
5. **hint**: A short text description that appears when hovering over the checkbox.
6. **iconSize**: Specifies the size of the checkbox's icon.
7. **activeStateEnabled**: Enables/disables the active state, which occurs when the checkbox is focused.
8. **focusStateEnabled**: Enables/disables the focus state for the checkbox.
9. **hoverStateEnabled**: Enables/disables the hover state for the checkbox when it is hovered over.
10. **isValid**: A Boolean value that determines if the checkbox is valid based on the validation rules.
11. **name**: The name attribute of the checkbox element.
12. **readOnly**: Makes the checkbox read-only, preventing user interaction.
13. **rtlEnabled**: Specifies if the checkbox supports right-to-left (RTL) text direction.
14. **tabIndex**: Defines the tab order of the checkbox.
15. **validationStatus**: The current validation state of the checkbox (valid/invalid).
16. **validationErrors**: An array of validation error messages.
17. **visible**: Specifies whether the checkbox is visible on the page.
18. **width**: Defines the width of the checkbox component.
19. **accessKey**: Sets a keyboard shortcut to focus the checkbox.
20. **elementAttr**: Allows customization of the checkbox’s HTML element attributes.

**Events:**

1. **onValueChanged**: Triggered when the checkbox's value (checked or unchecked) changes.
2. **onOptionChanged**: Fires when an option value is changed, such as enabling/disabling the checkbox.
3. **onInitialized**: Fires when the checkbox is initialized and the component is ready to use.
4. **onContentReady**: Triggered when the checkbox content is fully loaded and rendered.
5. **onDisposing**: Fires when the checkbox is about to be disposed and removed from the DOM.

**Methods:**

1. **option()**: Used to get or set an option of the checkbox.
2. **beginUpdate()**: Starts a batch update of the checkbox options, optimizing performance during multiple changes.
3. **endUpdate()**: Ends a batch update and applies all changes made using beginUpdate().
4. **dispose()**: Disposes of the checkbox component, cleaning up any associated resources.
5. **repaint()**: Forces the checkbox to re-render and update the UI.
6. **reset()**: Resets the checkbox value to its default state (unchecked).
7. **resetOption()**: Resets a specific option to its initial value.
8. **element()**: Returns the jQuery element of the checkbox in the DOM.
9. **focus()**: Focuses the checkbox element.
10. **registerKeyHandler()**: Registers a custom key event handler for the checkbox.
11. **off()**: Unsubscribes from an event for the checkbox.
12. **on()**: Subscribes to an event for the checkbox.
13. **Datebox:**

**Options:**

1. **value**: The current value of the DateBox, typically a Date object or a string representing the date.
2. **min**: The minimum date value that can be selected in the DateBox.
3. **max**: The maximum date value that can be selected in the DateBox.
4. **displayFormat**: The format in which the date will be displayed in the input field.
5. **type**: Defines the type of DateBox (e.g., "date", "datetime", "time").
6. **opened**: A Boolean that indicates whether the popup calendar is open.
7. **readOnly**: Makes the DateBox read-only, preventing user modification.
8. **disabled**: Disables the DateBox, preventing interaction.
9. **showClearButton**: A Boolean that determines whether to show the clear button to clear the selected value.
10. **placeholder**: A placeholder text shown when the DateBox is empty.
11. **useMaskBehavior**: Defines whether to use a mask (input masking) for the value in the DateBox.
12. **showAnalogClock**: A Boolean that determines whether to display an analog clock when selecting time.
13. **show24HourFormat**: Specifies whether the time format should be in 24-hour format.
14. **focusStateEnabled**: Enables or disables the focus state for the DateBox.
15. **activeStateEnabled**: Enables or disables the active state for the DateBox.
16. **tabIndex**: Defines the tab order for the DateBox.
17. **validationStatus**: The current validation status (valid/invalid).
18. **validationErrors**: An array of validation error messages.
19. **hint**: A text shown when the user hovers over the DateBox, providing additional information.
20. **elementAttr**: Customizes the HTML attributes of the DateBox element.
21. **rtlEnabled**: Specifies whether the DateBox supports right-to-left (RTL) text direction.
22. **width**: Specifies the width of the DateBox component.
23. **height**: Specifies the height of the DateBox component.
24. **name**: The name attribute for the DateBox input element.
25. **displayExpr**: Defines the expression used to display the value in the DateBox (e.g., when binding to an object).
26. **valueExpr**: Defines the expression used to set the value of the DateBox when binding to an object.

**Events:**

1. **onValueChanged**: Fired when the selected date or time value changes.
2. **onOpened**: Fired when the DateBox popup calendar is opened.
3. **onClosed**: Fired when the DateBox popup calendar is closed.
4. **onFocusIn**: Triggered when the DateBox input field gains focus.
5. **onFocusOut**: Triggered when the DateBox input field loses focus.
6. **onInitialized**: Fired when the DateBox component is initialized.
7. **onOptionChanged**: Fired when an option of the DateBox is changed.
8. **onContentReady**: Triggered when the DateBox content is fully loaded and rendered.
9. **onDisposing**: Fired when the DateBox component is about to be disposed and removed from the DOM.

**Methods:**

1. **option()**: Used to get or set an option of the DateBox.
2. **reset()**: Resets the value of the DateBox to its default state.
3. **resetOption()**: Resets a specific option to its initial value.
4. **beginUpdate()**: Begins a batch update, improving performance when multiple options are being updated.
5. **endUpdate()**: Ends a batch update and applies all changes made using beginUpdate().
6. **dispose()**: Disposes of the DateBox component and cleans up associated resources.
7. **repaint()**: Forces the DateBox to re-render and update the UI.
8. **element()**: Returns the jQuery element of the DateBox in the DOM.
9. **focus()**: Focuses the DateBox input field.
10. **selectText()**: Selects the text inside the DateBox input field.
11. **clear()**: Clears the current value in the DateBox input field.
12. **show()**: Shows the DateBox popup calendar.
13. **hide()**: Hides the DateBox popup calendar.
14. **Dropdownbox:**
15. **value: The currently selected value in the DropdownBox.**
16. **valueExpr: Defines the expression used to extract the value of each item from the data source.**
17. **displayExpr: Defines the expression used to display each item in the dropdown list.**
18. **dataSource: The data source for the DropdownBox, which can be an array, URL, or DataStore.**
19. **placeholder: The placeholder text shown when no value is selected.**
20. **searchEnabled: A Boolean that enables or disables the search functionality within the DropdownBox.**
21. **showClearButton: A Boolean that determines if the clear button is displayed to allow the user to clear the selection.**
22. **inputAttr: Allows customization of the input element’s attributes.**
23. **opened: A Boolean that specifies whether the dropdown is open or closed.**
24. **readOnly: Makes the DropdownBox read-only, preventing user selection.**
25. **disabled: Disables the DropdownBox, preventing any interaction.**
26. **clearButtonText: The text displayed on the clear button.**
27. **useSearch: Enables or disables the search functionality in the dropdown list.**
28. **grouped: A Boolean indicating whether the items in the dropdown should be grouped by a certain field.**
29. **groupTemplate: A custom template used to display group headers in the dropdown.**
30. **contentTemplate: A custom template for the dropdown content.**
31. **height: Specifies the height of the DropdownBox.**
32. **width: Specifies the width of the DropdownBox.**
33. **rtlEnabled: Enables or disables right-to-left (RTL) text direction.**
34. **tabIndex: Defines the tab order for the DropdownBox element.**
35. **validationStatus: The current validation state (valid/invalid).**
36. **validationErrors: An array of validation error messages.**
37. **showPopupButton: A Boolean that determines if the button to open the dropdown should be visible.**
38. **popupHeight: The height of the dropdown popup.**
39. **popupWidth: The width of the dropdown popup.**
40. **searchMode: Defines the search mode (e.g., "contains", "startswith") for filtering items in the dropdown.**
41. **minSearchLength: The minimum number of characters the user must type before search filtering is applied.**
42. **showTitle: A Boolean that determines whether the title of the dropdown should be displayed.**

**Events:**

1. **onValueChanged: Fired when the selected value in the DropdownBox changes.**
2. **onOpened: Fired when the dropdown is opened.**
3. **onClosed: Fired when the dropdown is closed.**
4. **onContentReady: Triggered when the content of the DropdownBox is fully loaded and rendered.**
5. **onDisposing: Fired when the DropdownBox is about to be disposed and removed from the DOM.**
6. **onOptionChanged: Fired when an option of the DropdownBox is changed.**
7. **onInitialized: Fired when the DropdownBox is initialized and ready to use.**
8. **onSelectionChanged: Triggered when the selected item in the dropdown changes.**
9. **onSearch: Fired when the search query is typed by the user in the search box.**

**Methods:**

1. **option(): Gets or sets an option of the DropdownBox.**
2. **reset(): Resets the DropdownBox to its default state.**
3. **resetOption(): Resets a specific option to its initial value.**
4. **beginUpdate(): Begins a batch update to optimize performance during multiple changes.**
5. **endUpdate(): Ends a batch update and applies all changes made using beginUpdate().**
6. **dispose(): Disposes of the DropdownBox component, releasing resources.**
7. **repaint(): Forces the DropdownBox to re-render and update the UI.**
8. **element(): Returns the jQuery element of the DropdownBox in the DOM.**
9. **focus(): Focuses the input field of the DropdownBox.**
10. **selectText(): Selects the text inside the input field of the DropdownBox.**
11. **clear(): Clears the current value in the DropdownBox.**
12. **show(): Opens the dropdown list.**
13. **hide(): Closes the dropdown list.**
14. **open(): Opens the dropdown list programmatically.**
15. **close(): Closes the dropdown list programmatically.**
16. **Numberbox:**

**Options:**

1. **value**: The current value of the NumberBox.
2. **valueExpr**: Defines the expression used to extract the value of each item from the data source (for bound data sources).
3. **displayExpr**: Defines the expression used to display each item from the data source.
4. **min**: The minimum value that can be selected in the NumberBox.
5. **max**: The maximum value that can be selected in the NumberBox.
6. **step**: Defines the increment/decrement value for the number when using the spin buttons.
7. **placeholder**: The placeholder text shown when the input field is empty.
8. **showSpinButtons**: A Boolean that determines whether the spin buttons are shown for incrementing/decrementing the value.
9. **format**: Specifies the format of the value, including options like decimal precision, currency, percentage, etc.
10. **mode**: Specifies the input mode (e.g., "text", "number").
11. **useLargeSpinButtons**: A Boolean that determines whether to display larger spin buttons for the NumberBox.
12. **showClearButton**: A Boolean that determines if a clear button is displayed to allow clearing the input field.
13. **inputAttr**: Customizes the HTML attributes of the NumberBox input element.
14. **disabled**: Disables the NumberBox, preventing any interaction.
15. **readOnly**: Makes the NumberBox read-only, preventing user modification.
16. **rtlEnabled**: Enables or disables right-to-left (RTL) text direction.
17. **tabIndex**: Defines the tab order for the NumberBox element.
18. **width**: Specifies the width of the NumberBox component.
19. **height**: Specifies the height of the NumberBox component.
20. **validationStatus**: The current validation state (valid/invalid).
21. **validationErrors**: An array of validation error messages.

**Events:**

1. **onValueChanged**: Fired when the value of the NumberBox changes.
2. **onInput**: Triggered when the user types input into the NumberBox.
3. **onFocusIn**: Fired when the NumberBox input field gains focus.
4. **onFocusOut**: Fired when the NumberBox input field loses focus.
5. **onKeyDown**: Triggered when a key is pressed while the NumberBox is focused.
6. **onKeyUp**: Fired when a key is released while the NumberBox is focused.
7. **onContentReady**: Triggered when the NumberBox content is fully loaded and rendered.
8. **onDisposing**: Fired when the NumberBox is about to be disposed and removed from the DOM.
9. **onInitialized**: Fired when the NumberBox is initialized and ready to use.
10. **onOptionChanged**: Fired when an option of the NumberBox is changed.

**Methods:**

1. **option()**: Gets or sets an option of the NumberBox.
2. **reset()**: Resets the value of the NumberBox to its default state.
3. **resetOption()**: Resets a specific option to its initial value.
4. **beginUpdate()**: Starts a batch update, optimizing performance when multiple options are being updated.
5. **endUpdate()**: Ends a batch update and applies all changes made using beginUpdate().
6. **dispose()**: Disposes of the NumberBox component and cleans up associated resources.
7. **repaint()**: Forces the NumberBox to re-render and update the UI.
8. **element()**: Returns the jQuery element of the NumberBox in the DOM.
9. **focus()**: Focuses the input field of the NumberBox.
10. **clear()**: Clears the value of the NumberBox input field.
11. **show()**: Makes the NumberBox visible (useful if the component is hidden).
12. **hide()**: Hides the NumberBox (useful for dynamic visibility control).
13. **Selectbox:**

**Options:**

1. **value**: The currently selected value in the SelectBox.
2. **dataSource**: The data source for the SelectBox, which can be an array, URL, or DataStore.
3. **displayExpr**: Defines the expression used to display each item from the data source.
4. **valueExpr**: Defines the expression used to extract the value of each item from the data source.
5. **placeholder**: The placeholder text displayed when the SelectBox is empty.
6. **searchEnabled**: A Boolean that enables or disables the search functionality inside the SelectBox.
7. **searchMode**: Defines the search mode for filtering items (e.g., "contains", "startswith").
8. **searchTimeout**: The delay time in milliseconds before the search is triggered after the user starts typing.
9. **minSearchLength**: The minimum number of characters the user must type before search filtering is applied.
10. **itemTemplate**: A custom template for rendering the items in the SelectBox.
11. **grouped**: A Boolean that indicates whether the items should be grouped by a certain field.
12. **groupTemplate**: A custom template for rendering group headers in the dropdown.
13. **showClearButton**: A Boolean that determines whether the clear button is displayed to allow clearing the selection.
14. **readOnly**: Makes the SelectBox read-only, preventing user interaction.
15. **disabled**: Disables the SelectBox, making it unclickable.
16. **showDataBeforeSearch**: A Boolean that specifies whether data should be shown before performing a search.
17. **inputAttr**: Allows customization of the input element’s attributes.
18. **clearButtonText**: The text displayed on the clear button.
19. **showSelectionControls**: A Boolean that determines whether selection controls (checkboxes) are shown.
20. **dropDownButtonTemplate**: A custom template for the dropdown button.
21. **width**: Specifies the width of the SelectBox component.
22. **height**: Specifies the height of the SelectBox component.
23. **validationStatus**: The current validation status (valid/invalid).
24. **validationErrors**: An array of validation error messages.
25. **hint**: A text shown when the user hovers over the SelectBox to provide additional information.
26. **tabIndex**: Defines the tab order for the SelectBox.
27. **itemHoldTimeout**: The duration in milliseconds before an item is considered held by the user.
28. **deferRendering**: A Boolean that delays rendering the dropdown until it is needed, improving performance.
29. **openOnFieldClick**: A Boolean that specifies whether the dropdown should open when the field is clicked.
30. **opened**: A Boolean that indicates whether the dropdown is open.
31. **dropDownOptions**: Custom settings for the dropdown popup (e.g., position, visibility).
32. **popupHeight**: The height of the dropdown popup.
33. **popupWidth**: The width of the dropdown popup.
34. **useItemTextAsTitle**: A Boolean that specifies whether to use the item's text as the title when selecting an option.

**Events:**

1. **onValueChanged**: Fired when the selected value in the SelectBox changes.
2. **onOpened**: Triggered when the dropdown is opened.
3. **onClosed**: Fired when the dropdown is closed.
4. **onItemClick**: Triggered when an item in the dropdown list is clicked.
5. **onItemContextMenu**: Fired when the user right-clicks on an item in the dropdown.
6. **onItemHold**: Triggered when an item in the dropdown is held for a specified duration.
7. **onContentReady**: Triggered when the SelectBox content is fully loaded and rendered.
8. **onDisposing**: Fired when the SelectBox is about to be disposed and removed from the DOM.
9. **onInitialized**: Fired when the SelectBox is initialized and ready to use.
10. **onOptionChanged**: Fired when an option of the SelectBox is changed.

**Methods:**

1. **option()**: Gets or sets an option of the SelectBox.
2. **reset()**: Resets the SelectBox value to its default state.
3. **resetOption()**: Resets a specific option to its initial value.
4. **beginUpdate()**: Starts a batch update to optimize performance when multiple options are being updated.
5. **endUpdate()**: Ends a batch update and applies all changes made using beginUpdate().
6. **dispose()**: Disposes of the SelectBox component and cleans up associated resources.
7. **repaint()**: Forces the SelectBox to re-render and update the UI.
8. **element()**: Returns the jQuery element of the SelectBox in the DOM.
9. **focus()**: Focuses the SelectBox input field.
10. **clear()**: Clears the value of the SelectBox input field.
11. **show()**: Makes the SelectBox visible (useful for dynamic visibility control).
12. **hide()**: Hides the SelectBox (useful for dynamic visibility control).
13. **open()**: Opens the dropdown list programmatically.
14. **close()**: Closes the dropdown list programmatically

**6) Textarea:**

**Options:**

1. **accessKey**: Specifies the access key for the TextArea element, allowing users to navigate to the element via keyboard.
2. **activeStateEnabled**: A Boolean that enables or disables the active state styling for the TextArea.
3. **autoResizeEnabled**: A Boolean that enables or disables the auto-resizing of the TextArea based on the content.
4. **disabled**: Disables the TextArea, preventing user interaction.
5. **elementAttr**: Allows customization of the TextArea’s DOM element’s attributes.
6. **focusStateEnabled**: A Boolean that enables or disables the focus state styling for the TextArea.
7. **height**: Defines the height of the TextArea component.
8. **hint**: Text that appears when the user hovers over the TextArea, offering additional information.
9. **hoverStateEnabled**: A Boolean that enables or disables hover state styling for the TextArea.
10. **inputAttr**: Allows customization of the input element’s attributes.
11. **isValid**: A Boolean that indicates whether the TextArea is valid according to its validation rules.
12. **maxHeight**: Specifies the maximum height for the TextArea.
13. **maxLength**: Specifies the maximum number of characters that can be entered in the TextArea.
14. **minHeight**: Specifies the minimum height for the TextArea.
15. **name**: The name attribute of the TextArea element, typically used when submitting a form.
16. **placeholder**: The placeholder text displayed when the TextArea is empty.
17. **readOnly**: Makes the TextArea read-only, preventing user modification.
18. **rtlEnabled**: A Boolean that enables or disables right-to-left (RTL) text direction.
19. **spellCheck**: A Boolean that enables or disables spell checking for the TextArea.
20. **stylingMode**: Defines the styling mode for the TextArea, such as "outlined" or "filled."
21. **tabIndex**: Defines the tab order of the TextArea element.
22. **text**: The current text displayed in the TextArea.
23. **validationError**: The error message related to validation if the TextArea is invalid.
24. **validationErrors**: An array of validation error messages for the TextArea.
25. **validationMessageMode**: Specifies how validation messages are displayed (e.g., "auto", "always").
26. **validationStatus**: The current validation status of the TextArea (valid/invalid).
27. **value**: The value entered in the TextArea.
28. **valueChangeEvent**: Defines the event that triggers value change (e.g., "input", "change").
29. **visible**: A Boolean that controls the visibility of the TextArea.
30. **width**: Specifies the width of the TextArea component.

**Methods:**

1. **blur()**: Removes focus from the TextArea.
2. **focus()**: Sets focus on the TextArea.
3. **reset()**: Resets the TextArea to its initial value.
4. **select()**: Selects the text inside the TextArea.
5. **updateDimensions()**: Updates the dimensions of the TextArea, typically used after resizing or content changes.

**Events:**

1. **onChange**: Fired when the value of the TextArea changes.
2. **onContentReady**: Triggered when the TextArea content is fully loaded and rendered.
3. **onCopy**: Fired when the user copies text from the TextArea.
4. **onCut**: Fired when the user cuts text from the TextArea.
5. **onDisposing**: Fired when the TextArea is about to be disposed and removed from the DOM.
6. **onEnterKey**: Triggered when the user presses the Enter key in the TextArea.
7. **onFocusIn**: Fired when the TextArea input field gains focus.
8. **onFocusOut**: Fired when the TextArea input field loses focus.
9. **onInitialized**: Fired when the TextArea is initialized and ready to use.
10. **onInput**: Triggered when the user types input into the TextArea.
11. **onKeyDown**: Fired when a key is pressed while the TextArea is focused.
12. **onKeyPress**: Triggered when a key is pressed and before the input is entered in the TextArea.
13. **onKeyUp**: Fired when a key is released while the TextArea is focused.
14. **onOptionChanged**: Fired when an option of the TextArea is changed.
15. **onPaste**: Fired when the user pastes content into the TextArea.
16. **onValueChanged**: Fired when the value of the TextArea changes.

**7) Textbox:**

**Options:**

1. **accessKey**: Specifies the access key for the TextBox element, enabling navigation via the keyboard.
2. **activeStateEnabled**: A Boolean that enables or disables the active state styling for the TextBox.
3. **buttons**: Specifies custom buttons to appear inside the TextBox (e.g., clear button, custom icons).
4. **disabled**: Disables the TextBox, preventing user interaction.
5. **elementAttr**: Allows customization of the TextBox's DOM element's attributes.
6. **focusStateEnabled**: A Boolean that enables or disables the focus state styling for the TextBox.
7. **height**: Defines the height of the TextBox component.
8. **hint**: A text shown when the user hovers over the TextBox, providing additional information.
9. **hoverStateEnabled**: A Boolean that enables or disables hover state styling for the TextBox.
10. **inputAttr**: Allows customization of the input element’s attributes (e.g., maxlength).
11. **isValid**: A Boolean that indicates whether the TextBox is valid according to its validation rules.
12. **label**: Defines the label text associated with the TextBox.
13. **labelMode**: Specifies how the label is displayed (e.g., floating, outside).
14. **mask**: Defines a mask pattern for the input value, which ensures the input follows a specific format (e.g., phone number).
15. **maskChar**: Defines the character used to represent unfilled characters in the mask (e.g., an underscore).
16. **maskInvalidMessage**: The message shown when the input value doesn't match the mask.
17. **maskRules**: Defines custom rules for the mask (e.g., allowing only numeric input).
18. **maxLength**: Specifies the maximum number of characters allowed in the TextBox.
19. **mode**: Specifies the input mode for the TextBox (e.g., "text", "password").
20. **name**: The name attribute for the TextBox, typically used when submitting a form.
21. **placeholder**: The placeholder text displayed when the TextBox is empty.
22. **readOnly**: Makes the TextBox read-only, preventing the user from editing the value.
23. **rtlEnabled**: A Boolean that enables or disables right-to-left (RTL) text direction.
24. **showClearButton**: A Boolean that determines whether the clear button is displayed to allow clearing the TextBox.
25. **showMaskMode**: Defines when to show the mask (e.g., always, on focus).
26. **spellCheck**: A Boolean that enables or disables spell checking in the TextBox.
27. **stylingMode**: Defines the styling mode for the TextBox (e.g., "outlined", "filled").
28. **tabIndex**: Defines the tab order for the TextBox.
29. **text**: The current text displayed in the TextBox.
30. **useMaskedValue**: A Boolean that determines whether to use the masked value or the raw value.
31. **validationError**: The validation error message if the TextBox is invalid.
32. **validationErrors**: An array of validation error messages.
33. **validationMessageMode**: Defines how validation messages are displayed (e.g., "auto", "always").
34. **validationStatus**: The current validation status of the TextBox (valid/invalid).
35. **value**: The value entered in the TextBox.
36. **valueChangeEvent**: The event that triggers value changes (e.g., "input", "change").
37. **visible**: A Boolean that controls the visibility of the TextBox.
38. **width**: Specifies the width of the TextBox component.

**Methods:**

1. **blur()**: Removes focus from the TextBox.
2. **focus()**: Sets focus on the TextBox.
3. **getButton(name)**: Retrieves a button (like the clear button) from inside the TextBox by its name.
4. **reset()**: Resets the TextBox value to its initial state.
5. **select()**: Selects the text inside the TextBox.
6. **updateDimensions()**: Updates the dimensions of the TextBox, typically after resizing or content changes.

**Events:**

1. **onChange**: Fired when the value of the TextBox changes.
2. **onContentReady**: Triggered when the TextBox content is fully loaded and rendered.
3. **onCopy**: Fired when the user copies text from the TextBox.
4. **onCut**: Fired when the user cuts text from the TextBox.
5. **onDisposing**: Fired when the TextBox is about to be disposed and removed from the DOM.
6. **onEnterKey**: Triggered when the user presses the Enter key in the TextBox.
7. **onFocusIn**: Fired when the TextBox gains focus.
8. **onFocusOut**: Fired when the TextBox loses focus.
9. **onInitialized**: Fired when the TextBox is initialized and ready to use.
10. **onInput**: Triggered when the user types input into the TextBox.
11. **onKeyDown**: Fired when a key is pressed while the TextBox is focused.
12. **onKeyPress**: Triggered when a key is pressed but before the input is entered in the TextBox.
13. **onKeyUp**: Fired when a key is released while the TextBox is focused.
14. **onOptionChanged**: Fired when an option of the TextBox is changed.
15. **onPaste**: Fired when the user pastes content into the TextBox.
16. **onValueChanged**: Fired when the value of the TextBox changes.

**8) Button**

**Options:**

1. **accessKey**: Specifies the access key for the button, allowing users to activate it via keyboard.
2. **activeStateEnabled**: A Boolean that enables or disables active state styling when the button is pressed.
3. **disabled**: A Boolean that disables the button, preventing user interaction.
4. **elementAttr**: Allows customization of the button’s DOM element’s attributes.
5. **height**: Defines the height of the button.
6. **hint**: The text displayed when the user hovers over the button.
7. **hoverStateEnabled**: A Boolean that enables or disables hover state styling for the button.
8. **icon**: Specifies the icon to be displayed on the button.
9. **iconPosition**: Defines the position of the icon relative to the button text (e.g., "left", "right").
10. **onClick**: Event handler triggered when the button is clicked.
11. **stylingMode**: Defines the styling mode for the button (e.g., "outlined", "text").
12. **tabIndex**: Defines the tab order of the button.
13. **text**: The text displayed on the button.
14. **type**: Defines the button type (e.g., "normal", "submit").
15. **useSubmitBehavior**: A Boolean that enables or disables the button’s behavior as a submit button.
16. **validationGroup**: Defines the validation group for the button (used with form validation).

**Methods:**

1. **click()**: Programmatically triggers the button click event.
2. **reset()**: Resets the button state to its initial configuration.

**Events:**

1. **onClick**: Fired when the button is clicked.
2. **onContentReady**: Fired when the button content is fully loaded.
3. **onDisposing**: Fired when the button is about to be disposed.
4. **onInitialized**: Fired when the button is initialized.

**9) FileUploader**

**Options:**

1. **accept**: Specifies the types of files that can be uploaded.
2. **activeStateEnabled**: A Boolean that enables or disables the active state styling for the FileUploader.
3. **disabled**: Disables the FileUploader, preventing user interaction.
4. **elementAttr**: Allows customization of the DOM element’s attributes.
5. **height**: Specifies the height of the FileUploader.
6. **multiple**: A Boolean that determines whether multiple files can be uploaded at once.
7. **name**: The name attribute of the FileUploader, useful when submitting a form.
8. **uploadMode**: Defines the upload mode ("instantly", "useButtons").
9. **uploadUrl**: Specifies the URL to which the files will be uploaded.
10. **showFileList**: A Boolean that specifies whether the uploaded files list is shown.
11. **validationStatus**: The validation status of the FileUploader (valid/invalid).
12. **visible**: A Boolean that controls the visibility of the FileUploader.

**Methods:**

1. **upload()**: Starts the file upload process.
2. **abort()**: Aborts the file upload process.
3. **clear()**: Clears the file list.
4. **reset()**: Resets the file uploader state.

**Events:**

1. **onFileUploaded**: Fired when a file is successfully uploaded.
2. **onValueChanged**: Fired when the selected files change.
3. **onFileUploadError**: Triggered when a file upload fails.
4. **onContentReady**: Fired when the FileUploader content is fully loaded and ready.
5. **onDisposing**: Fired when the FileUploader is about to be disposed.

**10) Validator**

**Options:**

1. **validationRules**: Specifies the validation rules to be applied to the associated elements.
2. **adapter**: Allows customization of the validation process by providing an adapter object.
3. **onValidated**: A callback function fired after validation.
4. **validationGroup**: Defines the validation group for the Validator (used for grouping multiple form validations).
5. **isValid**: A Boolean indicating if the validation has passed for the associated elements.

**Methods:**

1. **validate()**: Initiates validation for all fields within the validator.
2. **reset()**: Resets the validation state.
3. **getValidationErrors()**: Returns the list of validation errors.

**Events:**

1. **onValidated**: Fired when validation completes, whether successful or with errors.
2. **onValidationError**: Fired when an error occurs during validation.

**RadioGroup**

**Options:**

1. **dataSource**: Specifies the data source (array, URL, or DataStore) for the radio buttons.
2. **value**: The currently selected value in the RadioGroup.
3. **disabled**: Disables the RadioGroup, making it unclickable.
4. **elementAttr**: Allows customization of the RadioGroup's DOM element’s attributes.
5. **itemTemplate**: Specifies a custom template for each radio button item.
6. **layout**: Defines the layout of the RadioGroup, either "horizontal" or "vertical".
7. **name**: The name attribute for the RadioGroup, typically used for form submission.
8. **valueExpr**: Specifies the field that holds the value of each item.
9. **displayExpr**: Defines the field that holds the display text of each item.
10. **stylingMode**: Defines the styling mode for the RadioGroup (e.g., "outlined", "filled").
11. **tabIndex**: Specifies the tab order for the RadioGroup.
12. **validationStatus**: The validation status of the RadioGroup (valid/invalid).

**Methods:**

1. **option()**: Gets or sets an option of the RadioGroup.
2. **reset()**: Resets the RadioGroup to its default value.
3. **select()**: Programmatically selects a value in the RadioGroup.
4. **focus()**: Focuses the RadioGroup input.

**Events:**

1. **onValueChanged**: Fired when the value of the RadioGroup changes.
2. **onItemClick**: Triggered when an item in the RadioGroup is clicked.
3. **onInitialized**: Fired when the RadioGroup is initialized and ready.
4. **onDisposing**: Fired when the RadioGroup is about to be disposed.

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

Data Layer:

**1. ArrayStore**

**Options:**

* data – The in-memory array of objects managed by the store.
* key – The unique key field in objects for CRUD operations.
* onModified – A callback function that executes when data is modified.
* errorHandler – Handles errors that occur during data operations.

**Methods:**

* load(options) – Fetches data with filtering, sorting, and paging applied.
* byKey(key) – Retrieves a specific record by key.
* insert(values) – Adds a new record to the store.
* update(key, values) – Modifies an existing record by key.
* remove(key) – Deletes a record by key.
* clear() – Removes all data from the store.

**2. DataSource**

**Options:**

* store – The underlying store (e.g., ArrayStore, CustomStore).
* filter – Defines filtering conditions.
* sort – Specifies sorting criteria.
* group – Groups data based on a specified field.
* pageSize – Defines the number of items per page.
* paginate – Enables or disables pagination.
* searchExpr – Specifies fields for searching.
* searchOperation – Defines the search operation (e.g., contains, startswith).
* searchValue – The value used for searching.

**Methods:**

* load() – Fetches data from the store.
* reload() – Refreshes the data source.
* filter(criteria) – Applies filtering conditions dynamically.
* sort(ordering) – Sorts the data dynamically.
* group(field) – Groups data by a specific field.
* totalCount() – Returns the total number of records.
* isLastPage() – Checks if the last page has been reached.

**3. CustomStore**

**Options:**

* load(options) – Defines how data is fetched from the server.
* byKey(key) – Retrieves a specific record by key.
* insert(values) – Sends an insert request to the server.
* update(key, values) – Sends an update request.
* remove(key) – Sends a delete request.
* cacheRawData – Enables caching of raw data.
* loadMode – Determines whether to load data in raw or processed form.
* onLoading – Callback triggered before loading data.
* onLoaded – Callback triggered after data is loaded.
* onInserting – Callback before inserting a new record.
* onInserted – Callback after inserting a new record.
* onUpdating – Callback before updating a record.
* onUpdated – Callback after updating a record.
* onRemoving – Callback before removing a record.
* onRemoved – Callback after removing a record.

**Methods:**

* load(options) – Fetches data from the server with filtering, sorting, and paging.
* byKey(key) – Retrieves a record based on its key.
* insert(values) – Calls the server to add a new record.
* update(key, values) – Updates a record on the server.
* remove(key) – Deletes a record from the server.

**4. LocalStore**

**Options:**

* name – The key under which data is stored in localStorage.
* data – Initial data stored in localStorage.
* immediate – Determines whether changes are saved immediately.

**Methods:**

* load() – Fetches data from localStorage.
* insert(values) – Adds a new record to localStorage.
* update(key, values) – Updates an existing record.
* remove(key) – Deletes a record from localStorage.
* clear() – Removes all data from localStorage.

**5. Query**

**Methods:**

* filter(criteria) – Filters data based on specified conditions.
* sort(field, order) – Sorts data in ascending or descending order.
* groupBy(field) – Groups data by a specific field.
* select(fields) – Selects specific fields from the dataset.
* aggregate(method, field) – Performs aggregation (sum, avg, count, etc.).
* toArray() – Returns the final processed data as an array.
* count() – Returns the number of records in the dataset.
* slice(skip, take) – Paginates data by skipping and taking a specified number of records.

DataGrid:

**Data Binding**

* **dataSource** – Defines the data source of the grid.
* **dateSerializationFormat** – Specifies the format for serializing date values.
* **getDataSource()** – Retrieves the grid's data source instance.

**Paging and Scrolling**

* **pager** – Configures the pagination controls.
* **paging** – Enables or disables paging.
* **scrolling** – Configures scrolling behavior (standard, virtual, infinite).
* **pageCount()** – Returns the total number of pages.
* **pageIndex()** – Gets or sets the current page index.
* **pageSize()** – Gets or sets the number of rows per page.

**Editing**

* **editing** – Enables editing mode (batch, row, cell, popup, etc.).
* **addRow()** – Adds a new row to the grid.
* **deleteRow()** – Deletes a row by key or index.
* **editRow()** – Puts a row into edit mode.
* **editCell()** – Puts a specific cell into edit mode.
* **saveEditData()** – Saves all pending changes.
* **cancelEditData()** – Cancels all pending changes.
* **cellValue()** – Gets or sets a cell’s value.
* **closeEditCell()** – Closes the currently edited cell.
* **hasEditData()** – Checks if there are unsaved changes.

**Data Validation**

* **errorRowEnabled** – Displays an error row if validation fails.

**Cascading Lookups**

* **customizeColumns** – Allows modifying column settings dynamically.

**Grouping**

* **groupPanel** – Enables the group panel for drag-and-drop grouping.
* **grouping** – Configures group-related settings.
* **expandAll()** – Expands all grouped rows.
* **collapseAll()** – Collapses all grouped rows.
* **expandRow()** – Expands a specific grouped row.
* **collapseRow()** – Collapses a specific grouped row.

**Filtering**

* **filterBuilder** – Enables advanced filtering with a UI-based filter builder.
* **filterBuilderPopup** – Configures the filter builder popup.
* **filterRow** – Enables filtering within column headers.
* **filterPanel** – Displays an interactive filter panel.
* **filterSyncEnabled** – Synchronizes filter states across UI components.
* **clearFilter()** – Clears all applied filters.
* **searchPanel** – Enables a search bar for quick filtering.
* **searchByText()** – Filters grid data based on a text query.
* **getCombinedFilter()** – Retrieves the current filter expression.

**Sorting**

* **sorting** – Enables or disables sorting.
* **clearSorting()** – Resets sorting on all columns.

**Selection**

* **selection** – Configures row selection (single, multiple, checkbox).
* **selectedRowKeys** – Holds the selected row keys.
* **selectAll()** – Selects all rows.
* **deselectAll()** – Deselects all rows.
* **selectRows()** – Selects specific rows by key.
* **selectRowsByIndexes()** – Selects rows by index.
* **deselectRows()** – Deselects specific rows by key.
* **getSelectedRowKeys()** – Retrieves the keys of selected rows.
* **getSelectedRowsData()** – Retrieves the data of selected rows.
* **isRowSelected()** – Checks if a row is selected.

**Columns**

* **columns** – Defines column configurations.
* **columnAutoWidth** – Automatically adjusts column width.
* **columnChooser** – Enables a UI for selecting visible columns.
* **columnFixing** – Allows fixing columns to the left or right.
* **columnHidingEnabled** – Enables automatic hiding of columns on small screens.
* **columnMinWidth** – Sets the minimum width of columns.
* **columnResizingMode** – Defines how column resizing works.
* **columnResizing** – Enables or disables column resizing.
* **columnCustomization** – Allows customizing column settings dynamically.
* **addColumn()** – Adds a new column to the grid.
* **deleteColumn()** – Removes a column from the grid.
* **getCellElement()** – Retrieves the HTML element of a specific cell.

**Appearance**

* **hoverStateEnabled** – Enables hover effects on rows.
* **rowAlternationEnabled** – Enables alternate row coloring.
* **showBorders** – Displays borders around grid elements.
* **showColumnHeaders** – Shows or hides column headers.
* **showColumnLines** – Displays lines between columns.
* **showRowLines** – Displays lines between rows.
* **wordWrapEnabled** – Enables or disables text wrapping.

**Templates**

* **rowTemplate** – Defines a custom row layout.
* **cellCustomization** – Enables custom cell rendering.
* **columnTemplate** – Allows customizing column content.

**Toolbar Customization**

* **toolbarCustomization** – Configures toolbar buttons and elements.

**Data Summaries**

* **summary** – Enables summary rows with aggregate functions.
* **gridSummaries** – Displays summaries at the bottom of the grid.
* **groupSummaries** – Displays summaries within groups.
* **customSummaries** – Allows defining custom summary functions.

**Master-Detail**

* **masterDetail** – Enables nested grids or custom detail views.
* **masterDetailView** – Configures the master-detail relationship.

**Export**

* **export** – Enables exporting grid data.
* **exportToExcel()** – Exports data to an Excel file.

**Adaptability**

* **adaptiveDetailRow** – Enables adaptive row details for small screens.
* **isAdaptiveDetailRowExpanded()** – Checks if an adaptive detail row is expanded.
* **collapseAdaptiveDetailRow()** – Collapses an adaptive detail row.
* **expandAdaptiveDetailRow()** – Expands an adaptive detail row.
* **gridAdaptabilityOverview** – Provides an overview of adaptability settings.
* **gridColumnsHidingPriority** – Sets priority for hiding columns on small screens.

**Demo with Custom Store and Concept of DataGrid**

* **stateStoring** – Saves and restores grid state.
* **state()** – Retrieves or sets the current grid state.

**Methods**

* **beginCustomLoading(message)** – Displays a loading indicator with a custom message.
* **endCustomLoading()** – Hides the custom loading indicator.
* **byKey(key)** – Retrieves a data object by its key.
* **cancelEditData()** – Cancels all unsaved changes made by the user in the grid.
* **cellValue(rowIndex, columnName?)** – Gets or sets the value of a cell in the given row and column.
* **clearFilter()** – Removes all filters applied to the grid.
* **clearSelection()** – Clears all selected rows across all pages.
* **closeEditCell()** – Closes the currently open cell editor (used in batch or cell edit modes).
* **deselectAll()** – Deselects all selected rows across all pages, depending on selection mode.
* **editCell(rowIndex, columnName)** – Puts the specified cell into edit mode.
* **navigateToRow(key)** – Scrolls to the row with the given key.
* **pageCount()** – Returns the total number of pages in the grid.
* **columnCount()** – Returns the number of columns in the grid.
* **columnOption(columnIndex, option?)** – Gets or sets options for a specific column.
* **getCombinedFilter()** – Returns the current filter criteria applied to the grid.
* **getDataSource()** – Retrieves the underlying data source of the grid.
* **getTotalSummaryValue(summaryItemName?)** – Returns the total summary value for a specific summary item.
* **state()** – Returns or sets the grid's state, including filters, sorting, and grouping.
* **totalCount()** – Returns the total number of records in the grid (returns -1 if paging is disabled)

**Events**

* **onCellHoverChanged(e)** – Triggered when the mouse pointer enters or leaves a cell.
* **onCellPrepared(e)** – Fires after a cell is created and allows customization of its appearance.
* **onContextMenuPreparing(e)** – Fires before the context menu appears when right-clicking a cell, allowing customization of menu items.
* **onDataErrorOccurred(e)** – Triggered when a data loading error occurs (e.g., network failure or invalid data).
* **onEditCanceled(e)** – Fires when a user cancels editing without saving changes.
* **onEditorPrepared(e)** – Fires when a cell editor is initialized, allowing customization of the editor.
* **onRowClick(e)** – Triggered when a user clicks on a row.
* **onCellClick(e)** – Fires when a user clicks on a specific cell.
* **onRowPrepared(e)** – Fires after a row is created and allows customization of its appearance.

**Menu**

**Options**

* **dataSource** – Defines the menu items.
* **displayExpr** – Specifies which field in the data source should be displayed.
* **items** – Defines the menu structure manually.
* **itemTemplate** – Customizes the menu item rendering.
* **orientation** – Sets the menu orientation (horizontal/vertical).
* **rtlEnabled** – Enables right-to-left support.
* **showFirstSubmenuMode** – Configures how the first submenu appears.
* **submenuDirection** – Defines the direction of submenus.
* **adaptivityEnabled** – Enables menu responsiveness.
* **animation** – Configures submenu animations.
* **cssClass** – Applies a custom CSS class.
* **hideSubmenuOnMouseLeave** – Hides submenus when the mouse leaves.
* **selectByClick** – Enables menu selection by clicking.
* **selectedItem** – Stores the currently selected menu item.

**Methods**

* **open()** – Opens a specific submenu.
* **close()** – Closes an open submenu.
* **selectItem()** – Selects a menu item.
* **unselectItem()** – Unselects a menu item.
* **option()** – Gets or sets an option dynamically.
* **getSelectedItem()** – Returns the currently selected item.
* **updateDimensions()** – Updates menu layout when the container size changes.

**Events**

* **onItemClick** – Fires when a menu item is clicked.
* **onItemRendered** – Fires after an item is rendered.
* **onSelectionChanged** – Fires when the selection changes.
* **onSubmenuShowing** – Fires before a submenu is shown.
* **onSubmenuShown** – Fires after a submenu is shown.
* **onSubmenuHiding** – Fires before a submenu is hidden.
* **onSubmenuHidden** – Fires after a submenu is hidden.

**TreeView**

**Options**

* **dataSource** – Binds the tree to a data source.
* **displayExpr** – Specifies which field in the data source should be displayed.
* **items** – Defines the tree structure manually.
* **searchEnabled** – Enables searching within the tree.
* **searchExpr** – Specifies the fields used for searching.
* **searchMode** – Sets the search mode (contains, startsWith, etc.).
* **selectionMode** – Defines how selection works (single, multiple).
* **showCheckBoxesMode** – Enables checkboxes for selection.
* **expandNodesRecursive** – Expands all child nodes when a parent is expanded.
* **expandAllEnabled** – Enables a button to expand/collapse all nodes.
* **parentIdExpr** – Specifies the field for parent-child relationships.
* **expandedExpr** – Determines whether a node should be expanded.
* **hasItemsExpr** – Indicates whether a node has children.
* **hoverStateEnabled** – Enables hover effects.
* **focusStateEnabled** – Enables keyboard navigation focus.
* **animationEnabled** – Enables or disables node expansion animation.

**Methods**

* **selectItem()** – Selects a tree node.
* **unselectItem()** – Unselects a tree node.
* **collapseItem()** – Collapses a tree node.
* **expandItem()** – Expands a tree node.
* **getSelectedNodes()** – Retrieves selected nodes.
* **getSelectedNodeKeys()** – Gets selected node keys.
* **getNodes()** – Retrieves all tree nodes.
* **getNodeByKey()** – Finds a node by its key.
* **refresh()** – Reloads the tree data.
* **scrollToItem()** – Scrolls to a specific item.
* **updateDimensions()** – Updates tree layout when the container size changes.

**Events**

* **onItemClick** – Fires when a tree item is clicked.
* **onItemExpanded** – Fires when an item is expanded.
* **onItemCollapsed** – Fires when an item is collapsed.
* **onItemRendered** – Fires after a tree item is rendered.
* **onSelectionChanged** – Fires when selection changes.
* **onContentReady** – Fires when the tree is fully loaded.

**Load Indicator**

A visual indicator that represents a loading state.

**Options**

* **indicatorSrc** – Defines a custom loading indicator image.
* **rtlEnabled** – Enables right-to-left layout support.
* **visible** – Controls visibility.

**Methods**

* **option()** – Gets or sets an option dynamically.
* **dispose()** – Destroys the instance.

**Events**

* **onContentReady** – Fires when the widget is fully loaded.

**Load Panel**

A loading screen overlay that prevents user interaction during processing.

**Options**

* **animation** – Configures show/hide animations.
* **container** – Specifies the container for the load panel.
* **delay** – Sets a delay before showing the panel.
* **hideOnOutsideClick** – Determines if clicking outside closes the panel.
* **message** – Displays a loading message.
* **position** – Sets the panel’s position.
* **shading** – Enables a background overlay.
* **shadingColor** – Specifies the shading color.
* **visible** – Controls visibility.

**Methods**

* **show()** – Displays the load panel.
* **hide()** – Hides the load panel.
* **toggle()** – Toggles visibility.
* **option()** – Gets or sets an option dynamically.

**Events**

* **onShown** – Fires when the panel is shown.
* **onHidden** – Fires when the panel is hidden.
* **onContentReady** – Fires when the panel is fully rendered.

**Popup**

A floating container for displaying dialogs, forms, or messages.

**Options**

* **animation** – Configures show/hide animations.
* **closeOnOutsideClick** – Determines if clicking outside closes the popup.
* **contentTemplate** – Defines custom content rendering.
* **dragEnabled** – Allows dragging.
* **fullScreen** – Displays the popup in full-screen mode.
* **height** – Sets height.
* **position** – Defines placement.
* **resizeEnabled** – Allows resizing.
* **shading** – Enables a background overlay.
* **shadingColor** – Sets the shading color.
* **showCloseButton** – Displays a close button.
* **visible** – Controls visibility.
* **width** – Sets width.

**Methods**

* **show()** – Opens the popup.
* **hide()** – Closes the popup.
* **toggle()** – Toggles visibility.
* **option()** – Gets or sets an option dynamically.

**Events**

* **onShown** – Fires when the popup is shown.
* **onHidden** – Fires when the popup is hidden.
* **onContentReady** – Fires when the popup is fully rendered.
* **onResize** – Fires when the popup is resized.
* **onResizeEnd** – Fires when resizing is finished.
* **onResizeStart** – Fires when resizing begins.

**Popover**

A small floating popup used for contextual messages or additional information.

**Options**

* **animation** – Configures show/hide animations.
* **closeOnOutsideClick** – Determines if clicking outside closes the popover.
* **contentTemplate** – Defines custom content rendering.
* **position** – Specifies placement relative to the target element.
* **target** – Defines the element that triggers the popover.
* **visible** – Controls visibility.
* **width** – Sets width.

**Methods**

* **show()** – Opens the popover.
* **hide()** – Closes the popover.
* **toggle()** – Toggles visibility.
* **option()** – Gets or sets an option dynamically.

**Events**

* **onShown** – Fires when the popover is shown.
* **onHidden** – Fires when the popover is hidden.
* **onContentReady** – Fires when the popover is fully rendered.

**Toast**

A lightweight notification message that disappears after a few seconds.

**Options**

* **animation** – Configures show/hide animations.
* **closeOnClick** – Closes the toast when clicked.
* **displayTime** – Sets how long the toast is visible.
* **message** – The text displayed in the toast.
* **position** – Defines where the toast appears.
* **type** – Sets the toast type (success, error, warning, etc.).
* **visible** – Controls visibility.

**Methods**

* **show()** – Displays the toast.
* **hide()** – Hides the toast.
* **toggle()** – Toggles visibility.
* **option()** – Gets or sets an option dynamically.

**Events**

* **onShown** – Fires when the toast appears.
* **onHidden** – Fires when the toast disappears.
* **onContentReady** – Fires when the toast is fully loaded.