Creatio Academy

Front-end development Freedom UI

Freedom UI page

Version 8.0

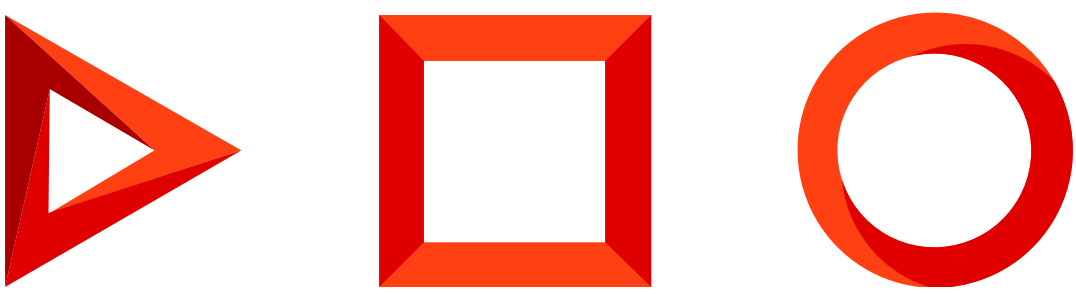


Table of Contents

Freedom UI page

Beginner

A **Freedom UI page** is an app element that lets you display Freedom UI elements connected to data sources and positioned arbitrarily. A [*Freedom UI page*] schema of the [*Client module*] type controls each Freedom UI page. Learn more in a separate article: Client module. For example, the `StudioHomePage` schema of the `UIv2` package configures the Studio home page. The `BaseTemplate` schema of the `UIv2` package implements the features of the base Freedom UI page. All form page schemas must inherit from `BaseTemplate` or its descendants. Freedom UI page examples: page with left area (the `PageWithLeftAreaTemplate` schema of the `UIv2` package), a page with a list (the `BaseGridSectionTemplate` schema of the `UIv2` package).

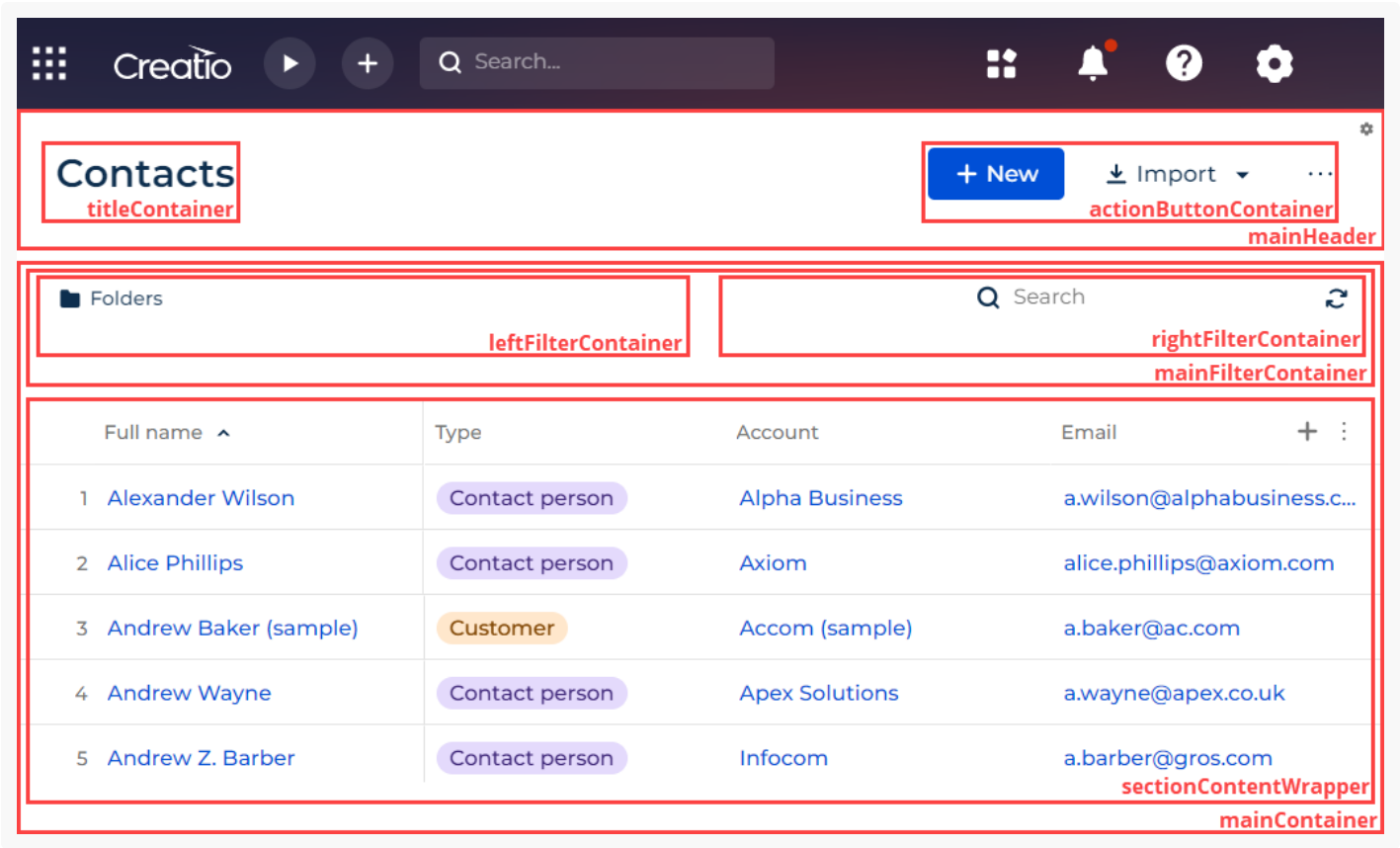
Depending on the current app template, an app can contain a list page and a form page with a minimum set of components. Customize the Freedom UI page further in the **Freedom UI Designer** using no-code tools. Learn more about setting up the Freedom UI Designer in the user documentation: Element setup examples.

Freedom UI page containers

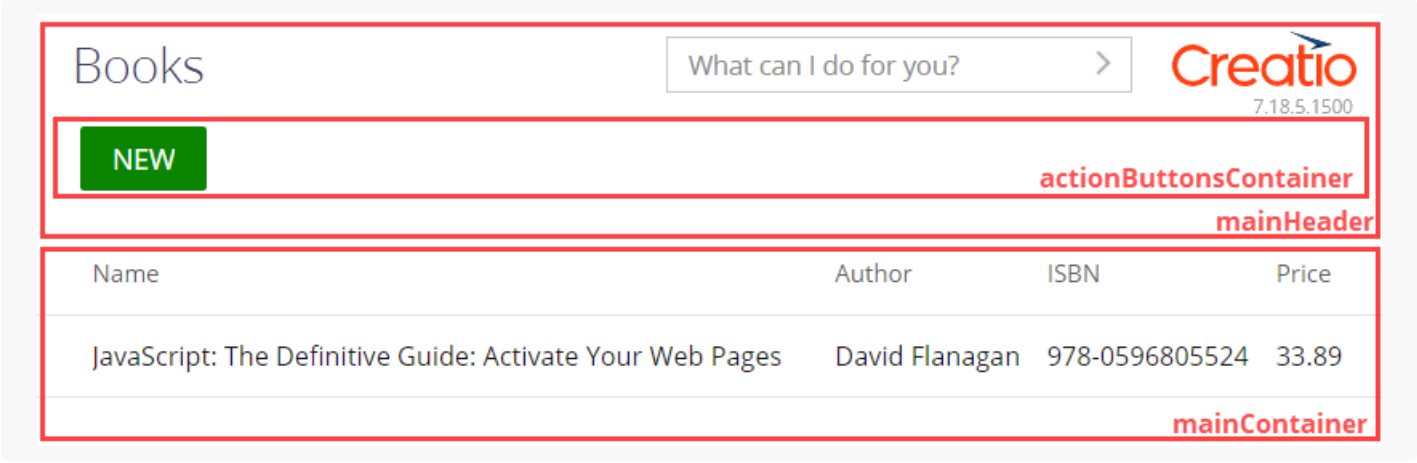
The Freedom UI page elements of the app are placed in the corresponding containers. Configure the containers using the base Freedom UI page schema or the replacing [*Freedom UI page*] schema. Containers operate similarly regardless of the page type.

> **Note.** Creatio uses HTML container meta names. It generates actual IDs that match the HTML elements of the form page based on the meta names.

We modified the main containers of the Freedom UI page in Creatio 8.0 Atlas. View the main **containers** of the Freedom UI page in Creatio version 8.0 and later in the figure below.



View the main **containers** of the Freedom UI page in Creatio version 7.18.5 in the figure below.



- The page header container (`mainHeader`). Includes the page title and the `actionButtonsContainer` child container.
- The page actions container (`actionButtonsContainer`). Includes the page actions, such as save, open, etc.
- The page content container (`mainContainer`). Includes the page content.

Learn more about creating a Freedom UI page in the user documentation: Set up the app UI.

Freedom UI page structure

Creatio Freedom UI includes the following page **structure elements**:

- **Data**. Learn more in the user documentation: Freedom UI Designer.
- **Charts**. Learn more in the user documentation: Freedom UI Designer.
- **Components** (button, list, label, folders, folder management menu, action dashboard). Learn more in the user documentation: Freedom UI Designer.
- **Layout elements**.
  - `FlexContainer` is a component that lets you lay out elements within it vertically or horizontally. The elements can have various sizes depending on their content. Built on top of `CSS Flexible Box`.
  - `GridContainer` is a component that lets you lay out its elements within it in a grid. The elements can have various sizes depending on their content. Built on top of `CSS Grid Layout`.

  Learn more in the user documentation: Freedom UI Designer.

FlexContainer component JS

Beginner

`FlexContainer` is a component that lets you lay out its children elements vertically or horizontally. The elements may have variable sizes depending on their content. Built on top of `CSS Flexible Box`.

The `FlexContainer` component lets you run the following **actions** on layout elements:

- Specify the element direction.
- Align elements.
- Distribute the space between elements.

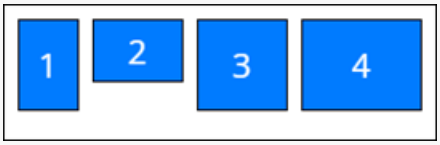
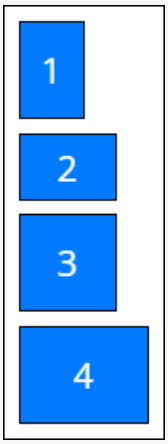
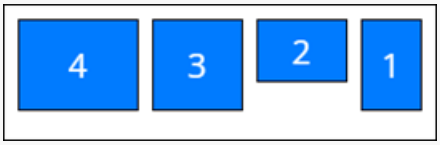
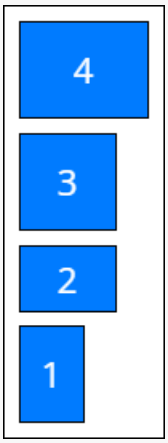
Learn more about `Flexible Box` in the following article: Basic concepts of flexbox.

Properties

`@Input direction`

Specifies the orientation and direction of the `flex` container's main axis to which `flex` elements align.

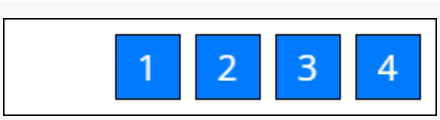
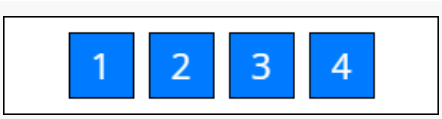
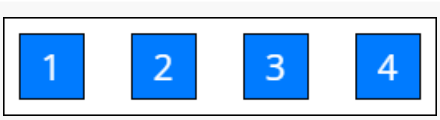
Available values

`row`	The children of the `flex` container align horizontally.	
`column`	The children of the `flex` container align vertically.	
`row-reverse`	The children of the `flex` container align horizontally in reverse.	
`column-reverse`	The children of the `flex` container align vertically in reverse.	

`@Input justifyContent`

Space distribution along the main axis. Defines how the browser distributes space between and around elements along the main axis of the `flex` container.

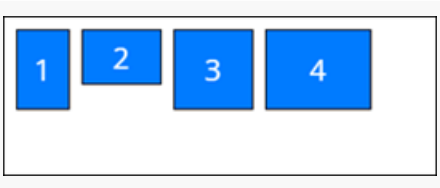
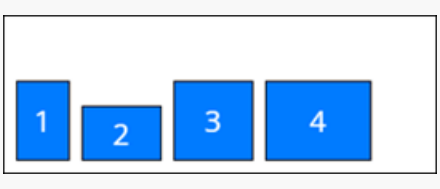
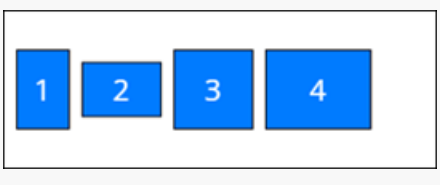
Available values

`start`	Pack elements from the start of the main `flex` container axis.	
`end`	Pack elements from the end of the main `flex` container axis.	
`center`	Pack elements around the center of the main `flex` container axis.	
`space-between`	The blocks of the `flex` container are distributed evenly. The first element is flush from the start of the main axis, the other is flush from the end.	

`@Input alignItems`

Space distribution along the cross axis. Defines how the browser distributes space between and around elements along the cross axis of the `flex` container.

Available values

`flex-start`	Pack elements from the start of the `flex` container's cross axis.	
`flex-end`	Pack elements from the end of the `flex` container's cross axis.	
`center`	Pack elements around the center of the main `flex` container axis.	

`@Input gap`

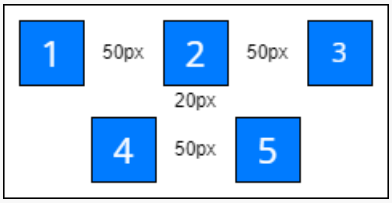
Defines the column and row spacing between the elements in the `flex` container. This is a shorthand for the `row-gap` (vertical gap) and `column-gap` (horizontal gap) properties.

View the resulting settings for `row-gap` and `column-gap` below.

Resulting settings for `row-gap` and `column-gap`

```
.container {
    display: flex;
    flex-wrap: wrap;
    column-gap: 50px;
    row-gap: 20px;
    justify-content: center;
}
```

View the resulting layout in the figure below.



`@Input padding`

Defines the container padding settings. This setting can apply a single value to all sides or provide a specific value for each side. The property can accept numbers, strings, and the available values listed below.

Available values

`none`	Zero padding.
`small`	Small padding.
`medium`	Medium padding.
`large`	Large padding.

View the example settings for the `padding` property below.

Example settings for the `padding` property

```
"padding": {
    "top": "6px",
```

```
    "right": 6,
    "bottom": "small",
    "left": "large"
}
```

`@Input border-radius`

Defines the corner rounding in the container. This setting can apply a single value to all corners or provide a specific value for each corner. The property can accept numbers, strings, and the available values listed below.

Available values

`none`	No rounding.
`small`	Small corner radius.
`medium`	Medium corner radius.
`large`	Large corner radius.

View the example settings for the `border-radius` property below.

Example settings for the `border-radius` property

```
"borderRadius": "medium"
```

`@Input color`

Defines the container color. This setting accepts a color code.

View the example settings for the `color` property below.

Example settings for the `color` property

```
"color": "#FDAB06"
```

Use example

An example of using the `FlexContainer` component in the Freedom UI page schema is available below.

Example of using the `FlexContainer` component in the Freedom UI page schema

```
"name": "FlexContainer",
"values": {
    "layoutConfig": {...},
    "type": "crt.FlexContainer",
    "direction": "column",
    "justifyContent": "start",
    "alignItems": "stretch",
    "wrap": "nowrap",
    "gap": "small",
    "items": []
}
...
{
    "operation": "insert",
    "name": "Button",
    "values": {
        "type": "crt.Button",
        "caption": "#ResourceString(Button_caption)#",
        "color": "primary",
    },
    "parentName": "FlexContainer",
    "propertyName": "Items",
    "index": 0
}
```

GridContainer component JS

Beginner

`GridContainer` is a component that lets you lay out its children elements within a grid. The elements may have variable size depending on their content. Built on top of `CSS Grid Layout`.

Learn more about `Grid Layout` in the following article: Basic Concepts of grid layout.

Properties

`@Input rows, columns`

Defines the track size of the layout grid.

Available values

Constant	The track size must be integer.

View the example for the `columns` and `rows` properties below.

Example settings for the `columns` and `rows` properties

```
{
    ...
    "columns": [
        "298px",
        "minmax(64px, 1fr)"
    ],
    "rows": "minmax(max-content, 32px)",
    ...
}
```

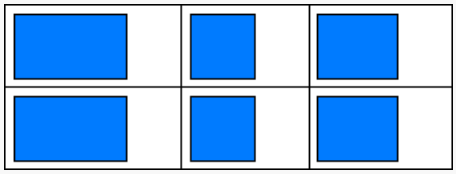
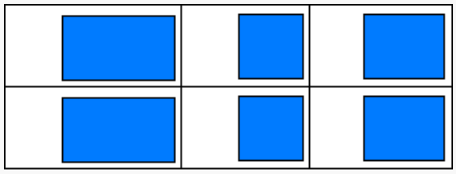
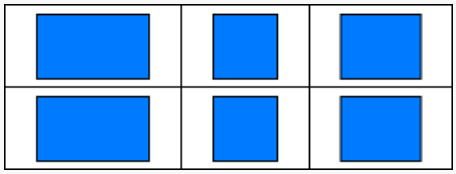
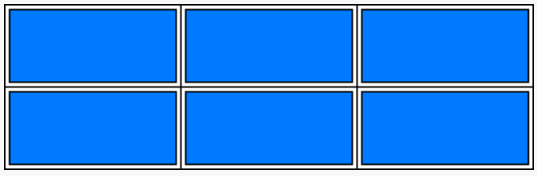
`GridContainer` does not support the following `CSS Grid Layout` **features**:

- The `grid-line-name` property. Required for named grid lines.
- The `fit-content()` function. Limits the size of a track. The function is the a `min(max-content, max(auto, argument))` formula calculated as the `auto` property (i. e., `minmax(auto, max-content)`). The track size is capped at the `argument` value as long as that value is larger than the minimum value of `auto`. In Creatio, the browser renders a single `32px`-wide column regardless of the argument in the `fit-content()` function.
- The `repeat()` function. Introduces a repeated fragment of a track list for compact representation of a recurring pattern.

`@Input justifyItems`

Space distribution along the main axis. Defines how the browser distributes space between and around elements relative to the horizontal axis of the `grid` container.

Available values

`start`	Pack the elements inside the block on the left.	
`end`	Pack the elements inside the block on the right.	
`center`	Pack the elements inside the block around the center.	
`stretch`	Pack the elements inside the block evenly from left to right.	

View the example settings for the `justify-items` property below.

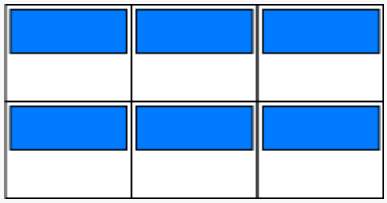
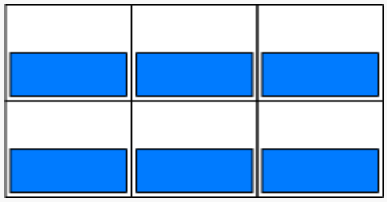
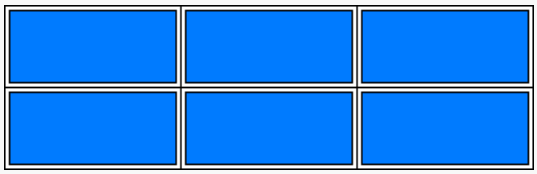
Example settings for the `justify-items` property

```
.container {
    justify-items: start | end | center | stretch;
}
```

`@Input alignItems`

Space distribution along the cross axis. Pack the elements inside the block vertically.

Available values

`start`	Pack the elements inside the block from top.	
`end`	Pack the elements inside the block at the bottom.	
`center`	Pack the elements inside the block around the center.	
`stretch`	Pack the elements inside the block evenly from top to bottom.	

View the example settings for the `align-items` property below.

Example settings for the `align-items` property

```
.container {
    align-items: start | end | center | stretch;
}
```

`@Input gap`

Defines the track spacing between the elements in the `grid` container. This is a shorthand for the `row-gap` (vertical gap) and `column-gap` (horizontal gap) properties.

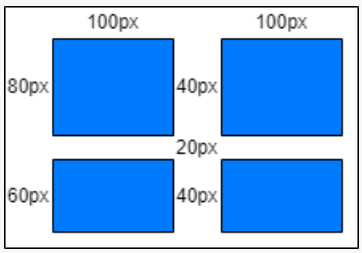
View the resulting settings for `row-gap` and `column-gap` below.

Resulting settings for `row-gap` and `column-gap`

```
.container {
```

```
    grid-template-columns: 100px auto;
    grid-template-rows: 80px 60px;
    column-gap: 40px;
    row-gap: 20px;
}
```

View the resulting layout in the figure below.



`@Input padding`

Defines the container padding settings. This setting can apply a single value to all sides or provide a specific value for each side. The property can accept numbers, strings, and the available values listed below.

Available values

`none`	Zero padding.
`small`	Small padding.
`medium`	Medium padding.
`large`	Large padding.

View the example settings for the `padding` property below.

Example settings for the `padding` property

```
"padding": {
    "top": "6px",
    "right": 6,
    "bottom": "small",
    "left": "large"
}
```

`@Input border-radius`

Defines the corner rounding in the container. This setting can apply a single value to all sides or provide a specific value for each side. The property can accept numbers, strings, and the available values listed below.

Available values

`none`	No rounding.
`small`	Small corner radius.
`medium`	Medium corner radius.
`large`	Large corner radius.

View the example settings for the `border-radius` property below.

Example settings for the `border-radius` property

```
"borderRadius": "medium"
```

`@Input color`

Defines the container color. This setting accepts a color code.

View the example settings for the `color` property below.

Example settings for the `color` property

```
"color": "#FDAB06"
```

Use example

An example of using the `GridContainer` component in the diagram is available below.

Example of using the `GridContainer` component in the diagram

```
"name": "GridContainer",
"values": {
    "layoutConfig": {...},
    "type": "crt.GridContainer",
    "columns": [
        "minmax(32px, 1fr)",
        "minmax(32px, 1fr)",
        "minmax(32px, 1fr)",
        "minmax(32px, 1fr)"
```

```
        ],
        "rows": "minmax(max-content, 32px)",
        "gap": {
            "columnGap": "large"
        },
        "items": []
    }
    ...
    {
        "operation": "insert",
        "name": "Button",
        "values": {
            "layoutConfig": {
                "column": 1,
                "row": 1,
                "colSpan": 3,
                "rowSpan": 1
            },
            "type": "crt.Button",
            "caption": "#ResourceString(Button_caption)#",
            "color": "default",
        },
        "parentName": "GridContainer",
        "propertyName": "Items",
        "index": 0
    }
```

DateTimePicker component JS

Beginner

`DateTimePicker` is a component that lets you set up a [*Date/Time*] type field on the app page. The `aria-label` attribute is generated automatically and refers to the `<input>` parent element.

Properties

`control` `FormControl`

Control.

`label` `string`

Label.

`ariaLabel` `string`

A property mapped to the `aria-label` element attribute.

`value` `string`

The value of the [*Date/Time*] type field.

`disabled` `boolean`

A flag that locks the [*Date/Time*] type field from editing.

`readonly` `boolean`

A flag that sets the [*Date/Time*] type field to read-only. By default, `false`.

`rowModeSizePx` `number`

The width of the [*Date/Time*] type field. By default, `320`.

`multiYearSelector` `boolean`

A flag that enables the year selector on year click. By default, `false`.

`twelvehour` `boolean`

If set to `true`, the field uses the 12-hour clock. By default, `false`.

`startView` `string`

The property that sets the period for the displayed data. By default, `month`.

Available values

`month`	Month.
`year`	Year.
`hour`	Hour.
`multi-year`	Several years.

`mode` `string`

The display mode of the [*Date/Time*] type field. By default, `auto`.

Available values

`auto`	Automatic mode.
`portrait`	Portrait mode.
`landscape`	Landscape mode.

`timeInterval` `number`

The search delay duration, in milliseconds. By default, `1`.

`preventSameDateTimeSelection` `boolean`

Lock selecting the currently selected date and time. By default, `false`.

Keyboard shortcuts

`LEFT_ARROW`

Highlight the previous day.

`RIGHT_ARROW`

Highlight the following day.

`UP_ARROW`

Highlight the previous week.

`DOWN_ARROW`

Highlight the next week.

`HOME`

Highlight the first day of the month.

`END`

Highlight the last day of the month.

`PAGE_UP`

Highlight the previous month.

`PAGE_DOWN`

Highlight the following month.

`ALT + PAGE_UP`

Highlight the previous year.

`ALT + PAGE_DOWN`

Highlight the following year.

`ENTER`

Select the highlighted element.

Checkbox component JS

Beginner

`Checkbox` is a component that lets you set up a checkbox on an app page.

Properties

`id` `string`

The unique ID. Service field.

`value` `boolean`

The flag that specifies the checkbox status. By default, `false`.

`disabled` `boolean`

The flag that locks the checkbox from editing. By default, `false`.

`inversed` `boolean`

The flag that inverts the checkbox style. By default, `false`.

`label` `string`

The checkbox title.

`ariaLabel` `string`

The checkbox property mapped to the `aria-label` element attribute. Learn more about the `aria-label` attribute in the official Mozilla documentation.

`labelPosition` `string`

Specifies where to display the checkbox title. By default, `auto`.

Available values

`auto`	Move the checkbox title automatically when the available screen space for the input changes. By default, to the left. When shrunk, above.
`left`	Display the title to the left of the checkbox.
`right`	Display the title to the right of the checkbox.
`hidden`	Hide the checkbox title.
`above`	Display the title above the checkbox.

Events

`valueChange` `boolean`

Fires when the checkbox value changes.

Input component JS

Beginner

`Input` is a component that lets you set up an input on an app page. The `Input` component uses its own `<input>` and `<label>` elements to ensure the the input is available by default. The `aria-label` attribute is generated automatically and refers to the `<input>` parent element.

Properties

`id` `string`

The unique ID. Service field.

`control` `FormControl`

The control.

`label` `string`

The input title.

`inputType` `string`

The input type. By default, `text`.

Available values

`text`	Sets the input type to text.
`password`	Sets the input type to text displayed using `*` characters. Includes a button to view the actual text.

`placeholder` `string`

The hint the input displays before you start filling it out.

`appearance` `enum`

The input appearance. By default, `legacy`.

Available values

`legacy`	Display only the bottom input border.
`outline`	Display all input borders.

`value` `string`

The input value.

`disabled` `boolean`

The flag that locks the input from editing.

`readonly` `boolean`

The flag that sets the input to read-only. By default, `false`.

`autocomplete` `string`

Permit the browser to populate the input automatically. By default, `off`.

`rowModeSizePx` `number`

The width of the input. When the width reaches the value and the `auto` mode is set, Creatio switches `Input` and `Checkbox` components from vertical to horizontal view. By default, `320`.

`labelPosition` `string`

Specifies where to display the input title. By default, `auto`.

Available values

`auto`	Move the input title automatically when the available screen space for the input changes. By default, to the left. When shrunk, above.
`left`	Display the title to the left of the input.
`right`	Display the title to the right of the input.
`hidden`	Hide the input title.
`above`	Display the title above the input.

NumberInput component JS

Beginner

`NumberInput` is a component that lets you set up a numeric field on an app page. The `NumberInput` component uses its own `<input>` and `<label>` elements to ensure the the field is available by default. The `aria-label` attribute is generated automatically and refers to the `<input>` parent element.

Properties

`id` `string`

The unique ID. Service field.

`control` `FormControl`

The control.

`label` `string`

The numeric field title.

`value` `string`

The numeric field value.

`disabled` `boolean`

The flag that locks the numeric field from editing.

`min` `number`

The minimum allowed numeric field value.

`max` `number`

The maximum allowed numeric field value.

`decimalPrecision` `number`

The number of decimal places.

`readonly` `boolean`

The flag that locks the numeric field from editing. By default, `false`.

`autocomplete` `string`

Permit the browser to populate the numeric field automatically. By default, `off`.

`rowModeSizePx` `number`

The numeric field width. By default, `320`.

`labelPosition` `string`

Specifies where to display the numeric field title. By default, `auto`.

Available values

`auto`	Move the numeric field title automatically when the available screen space for the input changes. By default, to the left. When shrunk, above.
`left`	Display the title to the left of the numeric field.
`right`	Display the title to the right of the numeric field.
`hidden`	Hide the numeric field title.
`above`	Display the title above the numeric field.

ImageInput component JS

Beginner

`ImageInput` is a component that lets you set up an image field on a Freedom UI page.

Properties

`id` `string`

The unique ID. Service field.

`value` `string`

The image field value. Can be bound to text or `ImageLink` attributes, such as, Base64, DataURI or CDN link to an image.

`readonly` `boolean`

Whether the field is read-only. By default, `false`.

`visible` `boolean`

Whether the field is visible. By default, `true`.

`positioning` `string`

The image position in the container. By default, `cover`.

Available values

`cover`	The image fills the entire element content box with possible cropping. Aspect ratio is preserved.
`scale-down`	The image fits the boundaries of the element content box without cropping. Aspect ratio is preserved.

`size` `string`

The image width and height.

Available values

`small`	32*32 px
`medium`	48*48 px
`large`	80*80 px
`extra-large`	120*120 px

`borderRadius` `string`

The type of corner rounding. By default, `medium`.

Available values

`none`	No rounding (0 px).
`small`	Small rounding (4 px).
`medium`	Medium rounding (8 px).
`large`	Large rounding (16 px).

`maxFileSize` `number`

The maximum allowed file size.

`customHeight` `string` **optional**

The custom image height in em, px, %, or other units supported by CSS. Fill out both `customHeight` and `customWidth` when you customize the dimensions.

`customWidth` `string` **optional**

The custom image width in em, px, %, or other units supported by CSS. Fill out both `customHeight` and `customWidth` when you customize the dimensions.

`placeholderMode` `ImagePlaceholderMode` **optional**

The mode of the placeholder. By default, `icon`.

Available values

`abbreviation`	Text placeholder.
`icon`	Icon placeholder.

`placeholder` `string` **optional**

Icon or text placeholder displayed if no image is added to the field. Must be bound to a text attribute if `placeholderMode` is set to `abbreviation`. Must be bound to an image from the repository if the `placeholderMode` is set to `icon`.

Dropdown component JS

Beginner

`Dropdown` is a component that lets you set up a drop-down list on an app page. The `Dropdown` component uses its own `<input>` and `<label>` elements to ensure the availability by default. The `aria-label` attribute is generated automatically and contains the label displayed using the `<label>` element.

Properties

`id` `string`

The unique ID. Service field.

`control` `FormControl`

Control.

`label` `string`

Label of the drop-down list.

`ariaLabel` `string`

A property mapped to the `aria-label` element attribute. Uses the `label` property value by default.

`value` `NullableLookupValue`

Object of the drop-down list element.

`disabled` `boolean`

The flag that locks the drop-down list from editing.

`readonly` `boolean`

The flag that sets the drop-down list to read-only. By default, `false`.

`items` `LookupValue[]`

Elements of the drop-down list. By default, `[]`.

`rowModeSizePx` `number`

The width of the drop-down list row. By default, `320`.

`debounceTime` `number`

The search delay duration, in milliseconds. By default, `500`.

ExpansionPanel component

Beginner

`ExpansionPanel` is a component that lets you set up an expansion panel on an app page. The `ExpansionPanel` uses its own `<button>` element to expand the content and `<label>` element to display the label and description.

Properties

`id` `string`

The unique ID. Service field.

`title` `string`

The expansion panel title.

`expanded` `boolean`

The expansion panel status. Set to `true` to expand the panel content, set to `false` to collapse the panel content.

`toggleType` `string`

The layout of the expander arrow. By default, `default`.

Available values

`default`	Red expander arrow with a background.
`material`	Blue expander arrow without a background.

`togglePosition` `string`

The position of the expander arrow. By default, `before`.

Available values

`before`	Display the arrow before the panel header.
`after`	Display the arrow after the panel header.

`ariaLabel` `string`

The text next to the expander arrow. By default, `title`.

`extraStyles` `object`

Additional styles that set up the header and expander arrow.

`description` `string`

The description of the expansion panel header.

`fullWidthHeader` `boolean`

The width of the expansion panel header. By default, `false`.

`titleWidth` `number`

The width of the expansion panel header in percentages. By default, `50`.

`labelColor` `string`

The color of the expansion panel title.

TabPanel component JS

Beginner

`TabPanel` is a component that lets you set up a tab panel on an app page.

Properties

`id` `string`

The unique ID. Service field.

`items` `TabItem[]`

The array of panel tabs. By default, `[]`.

`selectedTabIndex` `number`

Sets the selected panel tab. By default, `0`.

Button component JS

Beginner

`Button` is a component that implements the [*Button*] type component in the front-end. Use the [*Button*] type component to implement user interaction with a Freedom UI page.

Properties

`name` `string`

Button name. Required to bind the button to a handler, access the model attributes, etc.

`visible` `string`

Button visibility.

`color` `string`

Button style. By default, `default`.

Available values

`default`	Auxiliary white button.
`primary`	Primary. Blue button.
`accent`	Accent. Green button.
`warn`	Warning. Red button.

`ariaLabel` `string`

Button property mapped to the `aria-label` button attribute. Uses the `caption` property value by default.

`caption` `string`

Localizable button caption.

`disabled` `boolean`

Flag that locks the button. By default, `false` (unlocked).

`disableRipple` `boolean`

Property that disables the button animation on click. By default, `false` (animation enabled).

`textTransform` `string`

Display the style of the button caption.

Available values

`capitalize`	Capitalize the first letter of every word in the caption.
`lowercase`	Do not capitalize the first letter of every word in the caption.
`uppercase`	Capitalize every letter of the caption.
`none`	Do not convert the caption.
`inherit`	Inherit the styles of the parent element.

`clickMode` `ButtonClickMode`

Sets the button mode. By default, `default`.

Events

`clicked` `boolean`

Fires when a user clicks the button. Learn more in a separate article: Page customization.

Styles

`disabled-button-background`

Color of an inactive button.

`button-border-radius`

Button border radius.

Icon button

Icon button is an additional setting of a [*Button*] type component.

Properties

`iconPosition` `ButtonIconPositionEnum`

Position of the icon relative to the button caption. By default, `only-text`.

Available values

`only-text`		Do not display the icon. Display only the button caption.
`left-icon`		Display the icon to the left of the button caption.
`right-icon`		Display the icon to the right of the button caption.
`only-icon`		Display only the icon. Do not display the button caption.

`caption` `string`

Localizable schema caption displayed when holding the pointer over the button.

`icon` `string`

Icon to display next to the button caption.

Available values

`actions-button-icon`	
`add-button-icon`	
`back-button-icon`	

bars-button-icon	
calculator-button-icon	
close-button-icon	
delete-button-icon	
disk-warn-button-icon	
document-button-icon	
document-new-button-icon	
email-button-icon	
flag-button-icon	
gear-button-icon	
import-button-icon	
lock-button-icon	
message-warn-button-icon	
more-button-icon	
open-button-icon	
pencil-button-icon	
print-button-icon	

process-button-icon	
reload-button-icon	
save-button-icon	
settings-button-icon	
social-button-icon	

Menu-item component

`Menu-item` is a component that lets you add additional actions to a button menu item. Use the `Menu-item` component for any interaction that performs an action on the current page.

Properties

`caption` `string`

Localizable caption of the button menu item.

`visible` `boolean`

Flag that manages the menu item visibility.

`disabled` `boolean`

Flag that locks the menu item. By default, `false`.

`items` `CrtMenuItemViewElementConfig[]`

Index of button menu items. Displayed when you hold the pointer over the menu item that contains a submenu. The operation mechanism and properties of the button submenu and the button itself are identical.

`icon` `string`

Icon to display next to the menu item caption.

Available values

actions-button-icon	
add-button-icon	
back-button-icon	
bars-button-icon	
calculator-button-icon	
close-button-icon	
delete-button-icon	
disk-warn-button-icon	
document-button-icon	
document-new-button-icon	
email-button-icon	
flag-button-icon	
gear-button-icon	
import-button-icon	
lock-button-icon	
message-warn-button-icon	

more-button-icon	
open-button-icon	
pencil-button-icon	
print-button-icon	
process-button-icon	
reload-button-icon	
save-button-icon	
settings-button-icon	
social-button-icon	

`iconColor` `string`

Hex code of an icon color.