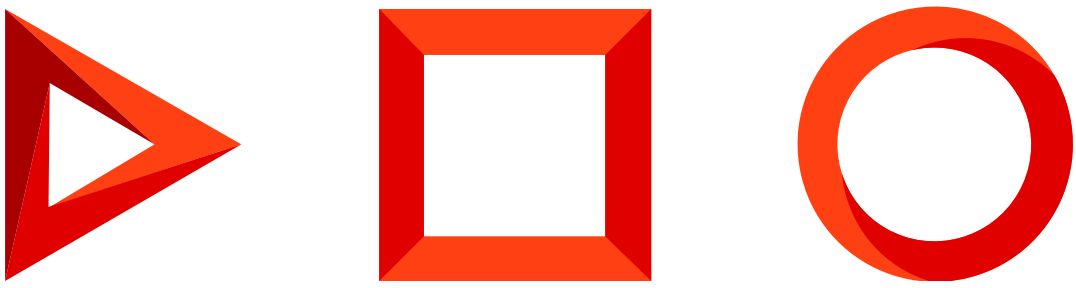
Creatio Academy

Страница

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

Страница

Основы

Страница — элемент приложения, который позволяет управлять внешним видом элементов, работать с источниками данных и произвольным образом размещать компоненты на странице. Каждая страница представлена схемой клиентского модуля. Например, домашняя страница Studio сконфигурирована в схеме `StudioHomePage` пакета `UIv2`. Функциональность базовой страницы реализована в схеме `BaseTemplate` пакета `UIv2`. Все схемы страниц записи должны наследовать страницу `BaseTemplate` или ее наследников. Примеры страницы: страница с островами слева (схема `PageWithLeftAreaTemplate` пакета `UIv2`), страница с реестром (схема `BaseGridSectionTemplate` пакета `UIv2`).

В зависимости от выбранного шаблона приложения, при создании приложение может содержать страницу с реестром и страницу записи с минимальным набором компонентов. Дальнейшая кастомизация выполняется в **дизайнере интерфейсов** с помощью no-code инструментов. Инструкция по настройке элементов в дизайнере интерфейсов содержится в статье Freedom UI дизайнер.

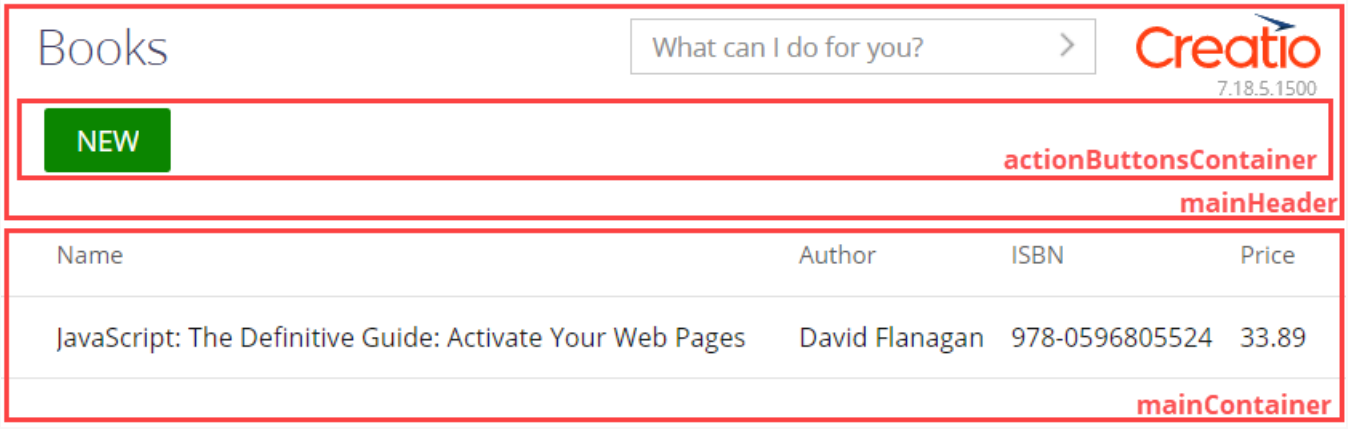
Кастомизация страницы описана в статье Кастомизация страницы.

Контейнеры страницы

Элементы пользовательского интерфейса приложения, которые относятся к странице, размещены в соответствующих контейнерах. Контейнеры конфигурируются в базовой схеме страницы или схеме замещающей страницы. Контейнеры не зависят от типа страницы.

> **На заметку.** В приложении используются мета-имена html-контейнеров. На основании мета-имен приложение формирует фактические идентификаторы соответствующих html-элементов страницы.

Основные **контейнеры** страницы представлены на рисунке ниже.



- Контейнер заголовка страницы (`mainHeader`) — заголовок страницы и вложенный контейнер

`actionButtonsContainer`.

- Контейнер действий страницы (`actionButtonsContainer`) — действия над страницей (например, сохранить, открыть и т. д.).
- Контейнер контента страницы (`mainContainer`) — контент страницы.

Создание страницы описано в статье Настроить приложение.

Структура страницы

Структурные элементы страницы в Creatio Freedom UI:

- **Данные**. Подробнее читайте в статье Freedom UI дизайнер.
- **Графики**. Подробнее читайте в статье Freedom UI дизайнер.
- **Компоненты** (кнопка, список, надпись, группы, меню управление группами, панель действий). Подробнее читайте в статье Freedom UI дизайнер.
- **Элементы разметки**.
  - `FlexContainer` — компонент разметки, который позволяет настроить расположение нескольких элементов последовательно в строку или колонку. Элементы могут изменять размер в зависимости от контента. Построен на базе компонента `CSS Flexible Box`.
  - `GridContainer` — компонент разметки, который позволяет настроить расположение нескольких элементов последовательно на сетке. Элементы могут изменять размер в зависимости от контента. Построен на базе `CSS Grid Layout`.

  Подробнее читайте в статье Freedom UI дизайнер.

Компонент FlexContainer JS

Основы

`FlexContainer` — компонент разметки, который позволяет настроить расположение нескольких элементов последовательно в строку или колонку. Элементы могут изменять размер в зависимости от контента. Построен на базе компонента `CSS Flexible Box`.

Действия, которые позволяет выполнять компонент `FlexContainer` с элементами макета:

- Задает направление элементов.
- Выравнивает элементы.
- Распределяет пространство между элементами.

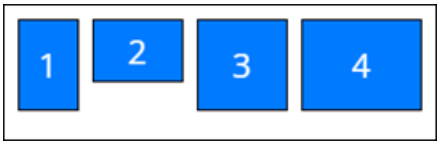
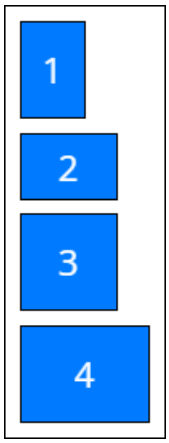
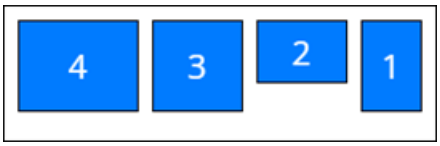
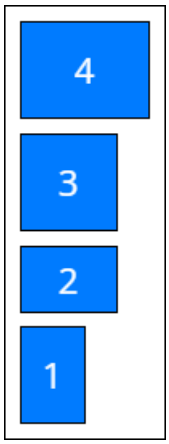
Подробно о `Flexible Box` читайте в статье Основные понятия Flexbox.

Свойства

```
@Input direction
```

Указывает как `flex`-элементы располагаются во `flex`-контейнере относительно главной оси и направления.

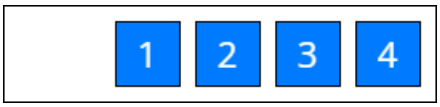
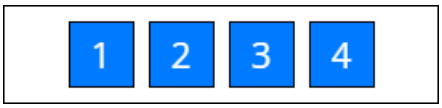
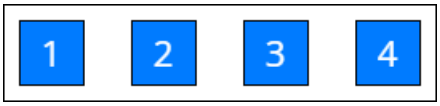
Возможные значения

`row`	Блоки `flex`-контейнера размещаются в строку.	
`column`	Блоки `flex`-контейнера размещаются в столбец.	
`row-reverse`	Блоки `flex`-контейнера размещаются в строку, но в обратном направлении.	
`column-reverse`	Блоки `flex`-контейнера размещаются в столбец, но в обратном направлении.	

`@Input justifyContent`

Выравнивание вдоль главной оси. Указывает как браузер распределяет пространство между и вокруг элементов контента вдоль главной оси `flex`-контейнера.

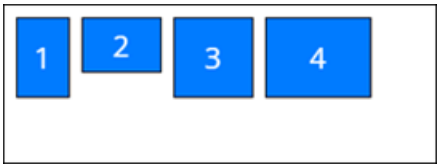
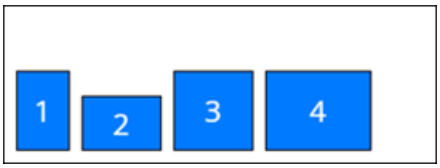
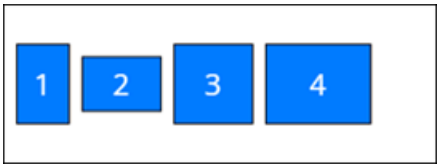
Возможные значения

`start`	Выравнивание элементов по левому краю `flex`-контейнера.	
`end`	Выравнивание элементов по правому краю `flex`-контейнера.	
`center`	Выравнивание элементов по центру `flex`-контейнера.	
`space-between`	Блоки `flex`-контейнера равномерно размещаются по всей ширине. Первый элемент выровнен по левому краю, последний — по правому.	

`@Input alignItems`

Выравнивание по вертикали. Выравнивает элементы `flex`-контейнера, как и свойство `@Input justifyContent`, но в перпендикулярном направлении.

Возможные значения

`flex-start`	Выравнивание элементов по верхнему краю `flex`-контейнера.	
`flex-end`	Выравнивание элементов по нижнему краю `flex`-контейнера.	
`center`	Выравнивание элементов по центру `flex`-контейнера.	

`@Input gap`

Задает отступы между элементами `flex`-контейнера в столбцах и строках. Является сокращением для свойств `row-gap` (отступ между элементами строки) и `column-gap` (отступ между элементами

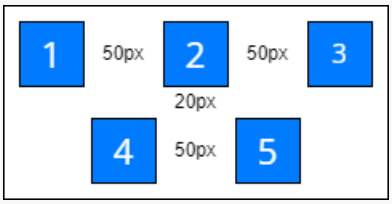
столбца).

Результат настройки свойств `row-gap` и `column-gap` приведен ниже.

Результат настройки свойств `row-gap` и `column-gap`

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

Результат отображения представлен на рисунке ниже.



`@Input padding`

Задает внутренние отступы для контейнера. Может настраиваться как для каждой стороны отдельно, так и для всех сторон одновременно. Может принимать число, строку и возможные значения, которые приведены ниже.

Возможные значения

`none`	Внутренний отступ отсутствует.
`small`	Маленький размер отступа.
`medium`	Средний размер отступа.
`large`	Большой размер отступа.

Пример настройки свойства `padding` приведен ниже.

Пример настройки свойства `padding`

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

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

`@Input border-radius`

Задает радиус скругления углов для контейнера. Может настраиваться как для каждой стороны отдельно, так и для всех сторон одновременно. Может принимать число, строку и возможные значения, которые приведены ниже.

Возможные значения

`none`	Скругление углов отсутствует.
`small`	Маленький радиус скругления углов.
`medium`	Средний радиус скругления углов.
`large`	Большой радиус скругления углов.

Пример настройки свойства `border-radius` приведен ниже.

Пример настройки свойства `border-radius`

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

`@Input color`

Задает цвет контейнера. В качестве значения принимает код цвета.

Пример настройки свойства `color` приведен ниже.

Пример настройки свойства `color`

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

Пример использования

Пример использования компонента `FlexContainer` в схеме представлен ниже.

Пример использования компонента `FlexContainer` в схеме

```
"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
}
```

Компонент GridContainer JS

Основы

`GridContainer` — компонент разметки, который позволяет настроить расположение нескольких элементов последовательно на сетке. Элементы могут изменять размер в зависимости от контента. Построен на базе `CSS Grid Layout`.

Подробно о `Grid Layout` читайте в статье Основные понятия Grid Layout.

Свойства

`@Input rows, columns`

Определяет ширину и высоту колонки сетки макета.

Возможные значения

Константа	Размер колонки задается целочисленным значением.

Пример настройки свойств `columns` и `rows` приведен ниже.

Пример настройки свойств `columns` и `rows`

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

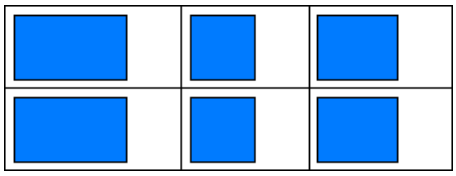
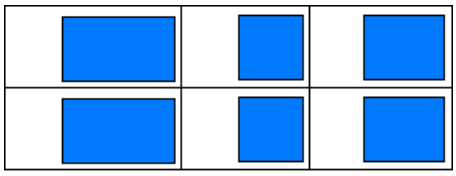
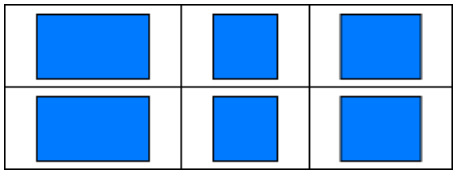
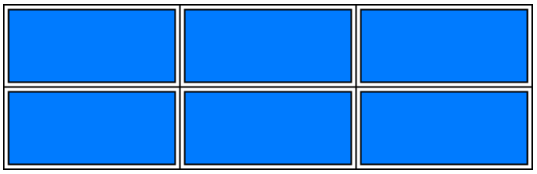
Свойства `CSS Grid Layout`, которые не поддерживает компонент `GridContainer`:

- `grid-line-name` — именованные линии.
- `fit-content()` — ограничивает ширину колонки. Представляет собой формулу `min(max-content, max(auto, argument))`, которая вычисляется, как свойство `auto` (т. е. `minmax(auto, max-content)`). Размер поля ограничен значением параметра `argument`, если он больше минимального значения, заданного в свойстве `auto`. В Creatio при любом заданном значении свойства `fit-content()` создается одна колонка шириной `32px`.
- `repeat()` — повторяющийся фрагмент перечня строк, который в компактной форме позволяет записать строки с повторяющимся шаблоном.

`@Input justifyItems`

Выравнивание по горизонтали. Указывает как браузер распределяет пространство между и вокруг элементов контента вдоль строчной оси `grid`-контейнера.

Возможные значения

`start`	Выравнивание элементов внутри блока по левому краю.	
`end`	Выравнивание элементов внутри блока по правому краю.	
`center`	Выравнивание элементов внутри блока по центру.	
`stretch`	Выравнивание элементов внутри блока по ширине.	

Пример настройки свойства `justify-items` приведен ниже.

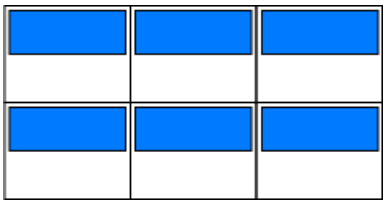
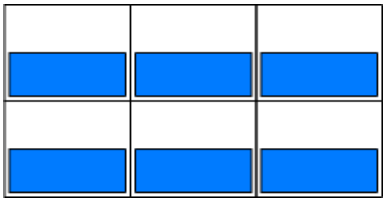
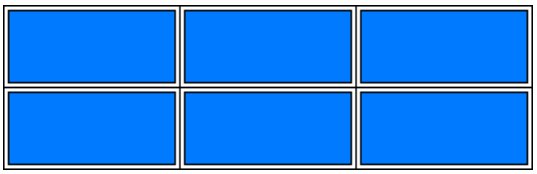
Пример настройки свойства `justify-items`

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

`@Input alignItems`

Выравнивание по вертикали. Выравнивание элементов внутри блока вдоль соответствующей оси.

Возможные значения

`start`	Выравнивание элементов внутри блока по верхнему краю.	
`end`	Выравнивание элементов внутри блока по нижнему краю.	
`center`	Выравнивание элементов внутри блока по центру.	
`stretch`	Выравнивание элементов внутри блока по ширине.	

Пример настройки свойства `align-items` приведен ниже.

Пример настройки свойства `align-items`

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

`@Input gap`

Задает отступы между элементами `grid`-контейнера в столбцах и строках. Является сокращением для свойств `row-gap` (отступ между элементами строки) и `column-gap` (отступ между элементами столбца).

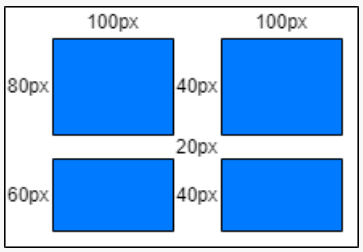
Результат настройки свойств `row-gap` и `column-gap` приведен ниже.

Результат настройки свойств `row-gap` и `column-gap`

```
.container {
    grid-template-columns: 100px auto;
    grid-template-rows: 80px 60px;
```

```
    column-gap: 40px;
    row-gap: 20px;
}
```

Результат отображения представлен на рисунке ниже.



`@Input padding`

Задает внутренние отступы для контейнера. Может настраиваться как для каждой стороны отдельно, так и для всех сторон одновременно. Может принимать число, строку и возможные значения, которые приведены ниже.

Возможные значения

`none`	Внутренний отступ отсутствует.
`small`	Маленький размер отступа.
`medium`	Средний размер отступа.
`large`	Большой размер отступа.

Пример настройки свойства `padding` приведен ниже.

Пример настройки свойства `padding`

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

`@Input border-radius`

Задает радиус скругления углов для контейнера. Может настраиваться как для каждой стороны отдельно, так и для всех сторон одновременно. Может принимать число, строку и возможные значения, которые приведены ниже.

Возможные значения

`none`	Скругление углов отсутствует.
`small`	Маленький радиус скругления углов.
`medium`	Средний радиус скругления углов.
`large`	Большой радиус скругления углов.

Пример настройки свойства `border-radius` приведен ниже.

Пример настройки свойства `border-radius`

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

`@Input color`

Задает цвет контейнера. В качестве значения принимает код цвета.

Пример настройки свойства `color` приведен ниже.

Пример настройки свойства `color`

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

Пример использования

Пример использования компонента `GridContainer` в схеме представлен ниже.

Пример использования компонента `GridContainer` в схеме

```
"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
    }
```

Компонент DateTimePicker JS

Основы

`DateTimePicker` — компонент, который позволяет настроить поле типа [*Дата/Время*] ([*Date/Time*]) на странице приложения. Атрибут `aria-label` генерируется автоматически и указывает на родительский элемент `<input>`.

Свойства

`id` `string`

Уникальный идентификатор. Служебное поле.

`control` `FormControl`

Элемент управления.

`label` `string`

Метка.

`ariaLabel` `string`

Свойство, которое связано с атрибутом `aria-label` элемента.

`value` `string`

Значение поля типа [*Дата/Время*] ([*Date/Time*]).

`disabled` `boolean`

Признак, который управляет доступностью изменения поля типа [*Дата/Время*] ([*Date/Time*]).

`readonly` `boolean`

Признак, который устанавливает доступность поля типа [*Дата/Время*] ([*Date/Time*]) только для чтения. По умолчанию — `false`.

`rowModeSizePx` `number`

Ширина поля типа [*Дата/Время*] ([*Date/Time*]). По умолчанию — `320`.

`multiYearSelector` `boolean`

Признак, который управляет возможностью просмотра списка годов при нажатии на год. По умолчанию — `false`.

`twelvehour` `boolean`

Если свойство установлено в `true`, то используется 12-часовой формат времени. По умолчанию — `false`.

`startView` `string`

Свойство, которое устанавливает период для просмотра данных. По умолчанию — `month`.

Возможные значения

`month`	Месяц.
`year`	Год.
`hour`	Час.
`multi-year`	Несколько лет.

`mode` `string`

Режим отображения поля типа [*Дата/Время*] ([*Date/Time*]). По умолчанию — `auto`.

Возможные значения

`auto`	Автоматический режим.
`portrait`	Вертикальный режим.
`landscape`	Горизонтальный режим.

`timeInterval` `number`

Длительность задержки поиска в миллисекундах. По умолчанию — `1`.

`preventSameDateTimeSelection` `boolean`

Запретить пользователю выбирать текущую выбранную дату и время. По умолчанию — `false`.

Горячие клавиши

`LEFT_ARROW`

Установить фокус на предыдущий день.

`RIGHT_ARROW`

Установить фокус на следующий день.

`UP_ARROW`

Установить фокус на предыдущую неделю.

`DOWN_ARROW`

Установить фокус на следующую неделю.

`HOME`

Установить фокус на первый день месяца.

`END`

Установить фокус на последний день месяца.

`PAGE_UP`

Установить фокус на предыдущий месяц.

`PAGE_DOWN`

Установить фокус на следующий месяц.

`ALT + PAGE_UP`

Установить фокус на предыдущий год.

`ALT + PAGE_DOWN`

Установить фокус на следующий год.

`ENTER`

Выбрать элемент, на котором установлен фокус.

Компонент Checkbox JS

Основы

`Checkbox` — компонент, который позволяет настроить чекбокс на странице приложения.

Свойства

`id` `string`

Уникальный идентификатор. Служебное поле.

`value` `boolean`

Признак, который устанавливает чекбокс. По умолчанию — `false`.

`disabled` `boolean`

Признак, который управляет доступностью изменения чекбокса. По умолчанию — `false`.

`inversed` `boolean`

Признак, который инвертирует стиль чекбокса. По умолчанию — `false`.

`label` `string`

Заголовок чекбокса.

`ariaLabel` `string`

Свойство чекбокса, которое связано с атрибутом `aria-label`. Описание атрибута `aria-label` содержится в официальной документации Mozilla.

`labelPosition` `string`

Устанавливает место отображения заголовка чекбокса. По умолчанию — `auto`.

Возможные значения

`auto`	Автоматический перенос заголовка чекбокса при изменении доступного места для элемента ввода на экране. По умолчанию — слева, при сжатии — сверху.
`left`	Заголовок отображается слева от чекбокса.
`right`	Заголовок отображается справа от чекбокса.
`hidden`	Заголовок чекбокса скрыт.
`above`	Заголовок отображается сверху чекбокса.

События

`valueChange` `boolean`

Событие, которое генерируется при изменении значения чекбокса.

Компонент NumberInput JS

Основы

`NumberInput` — компонент, который позволяет настроить числовое поле на странице приложения. Компонент `NumberInput` использует собственные элементы `<input>` и `<label>` для доступности поля по умолчанию. Атрибут `aria-label` генерируется автоматически и указывает на родительский элемент `<input>`.

Свойства

`id` `string`

Уникальный идентификатор. Служебное поле.

`control` `FormControl`

Элемент управления.

`label` `string`

Заголовок числового поля.

`value` `string`

Значение числового поля.

`disabled` `boolean`

Признак, который управляет доступностью изменения числового поля.

`min` `number`

Минимальное допустимое значение числового поля.

`max` `number`

Максимальное допустимое значение числового поля.

`decimalPrecision` `number`

Количество знаков после запятой.

`readonly` `boolean`

Признак, который устанавливает доступность изменения значения числового поля. По умолчанию — `false`.

`autocomplete` `string`

Устанавливает разрешения для браузера на автоматическое заполнение числового поля. По умолчанию — `off`.

`rowModeSizePx` `number`

Ширина числового поля. По умолчанию — `320`.

`labelPosition` `string`

Устанавливает место отображения заголовка числового поля. По умолчанию — `auto`.

Возможные значения

`auto`	Автоматический перенос заголовка числового поля при изменении доступного места для элемента ввода на экране. По умолчанию — слева, при сжатии — сверху.
`left`	Заголовок отображается слева от числового поля.
`right`	Заголовок отображается справа от числового поля.
`hidden`	Заголовок числового поля скрыт.
`above`	Заголовок отображается сверху числового поля.

Компонент Dropdown JS

Основы

`Dropdown` — компонент, который позволяет настроить выпадающий список на странице приложения. Компонент `Dropdown` использует собственные элементы `<input>` и `<label>`, чтобы обеспечить доступность по умолчанию. Атрибут `aria-label` генерируется автоматически и содержит заголовок, который отображен с помощью элемента `<label>`.

Свойства

`id` `string`

Уникальный идентификатор. Служебное поле.

`control` `FormControl`

Элемент управления.

`label` `string`

Заголовок выпадающего списка.

`ariaLabel` `string`

Свойство, которое связано с атрибутом `aria-label` элемента. По умолчанию используется значение свойства `label`.

`value` `NullableLookupValue`

Объект элемента выпадающего списка.

`disabled` `boolean`

Признак, который управляет доступностью изменения выпадающего списка.

`readonly` `boolean`

Признак, который устанавливает доступность выпадающего списка только для чтения. По умолчанию — `false`.

`items` `LookupValue[]`

Элементы выпадающего списка. По умолчанию — `[]`.

`rowModeSizePx` `number`

Ширина строки выпадающего списка. По умолчанию — `320`.

`debounceTime` `number`

Длительность задержки поиска в миллисекундах. По умолчанию — `500`.

Компонент ExpansionPanel JS

Основы

`ExpansionPanel` — компонент, который позволяет настроить группу на странице приложения. Компонент `ExpansionPanel` использует собственный элемент `<button>` для переключения содержимого и элемент `<label>` для отображения заголовка и описания.

Свойства

`id` `string`

Уникальный идентификатор. Служебное поле.

`title` `string`

Определяет заголовок группы.

`expanded` `boolean`

Определяет состояние группы. `true` — контент группы развернут, `false` — контент группы свернут.

`toggleType` `string`

Определяет стиль отображения иконки переключения группы. По умолчанию — `default`.

Возможные значения

`default`	Иконка переключения красного цвета с фоном.
`material`	Иконка переключения синего цвета без фона.

`togglePosition` `string`

Устанавливает место отображения иконки переключения. По умолчанию — `before`.

Возможные значения

`before`	Метка отображается перед заголовком группы.
`after`	Метка отображается после заголовка группы.

`ariaLabel` `string`

Определяет текст, который указан для переключения. По умолчанию — `title`.

`extraStyles` `object`

Определяет дополнительные стили для настройки заголовка и переключателя.

`description` `string`

Определяет описание заголовка группы.

`fullWidthHeader` `boolean`

Определяет ширину заголовка группы. По умолчанию — `false`.

`titleWidth` `number`

Определяет ширину заголовка группы в процентах. По умолчанию — `50`.

`labelColor` `string`

Цвет заголовка группы.

Компонент TabPanel JS

Основы

`TabPanel` — компонент, который позволяет настроить панель вкладок на странице приложения.

Свойства

`id` `string`

Уникальный идентификатор. Служебное поле.

`items` `TabItem[]`

Массив вкладок панели. По умолчанию — `[]`.

`selectedTabIndex` `number`

Устанавливает выбранную на панели вкладку. По умолчанию — `0`.

Компонент Button JS

Основы

`Button` — компонент, который реализует компонент типа [*Кнопка*] ([*Button*]) на front-end стороне. Компонент типа [*Кнопка*] ([*Button*]) используется для реализации взаимодействия пользователя со страницей Freedom UI.

Свойства

`name` `string`

Имя кнопки. Используется для привязки кнопки к обработчику, для обращения к атрибутам модели и т. д.

`visible` `string`

Видимость кнопки.

`color` `string`

Стиль кнопки. По умолчанию — `default`.

Возможные значения

`default`	Кнопка белого цвета.
`primary`	Основной. Кнопка синего цвета.
`accent`	Акцент. Кнопка зеленого цвета.
`warn`	Предупреждение. Кнопка красного цвета.

`ariaLabel` `string`

Свойство кнопки, которое связано с атрибутом `aria-label` кнопки. По умолчанию — значение свойства `caption`.

`caption` `string`

Локализуемый заголовок кнопки.

`disabled` `boolean`

Признак, который управляет блокировкой кнопки. По умолчанию — `false` (кнопка разблокирована).

`disableRipple` `boolean`

Свойство, которое управляет анимацией кнопки при ее нажатии. По умолчанию — `false` (анимация включена).

`textTransform` `string`

Стиль отображения заголовка кнопки.

Возможные значения

`capitalize`	Первая буква каждого слова заголовка отображается в верхнем регистре.
`lowercase`	Первая буква каждого слова заголовка отображается в нижнем регистре.
`uppercase`	Все буквы заголовка отображаются в верхнем регистре.
`none`	Конвертация заголовка не выполняется.
`inherit`	Наследует стили родительского элемента.

`clickMode` `ButtonClickMode`

Устанавливает режим кнопки. По умолчанию — `default`.

События

`clicked` `boolean`

Событие, которое генерируется при нажатии на кнопку. Подробнее читайте в статье Кастомизация страницы.

Стили

`disabled-button-background`

Цвет неактивной кнопки.

`button-border-radius`

Радиус закругления кнопки.

Кнопка с иконкой

Кнопка с иконкой — дополнительная настройка для компонента типа [*Кнопка*] ([*Button*]).

Свойства

`iconPosition` `ButtonIconPositionEnum`

Расположение иконки кнопки относительно ее заголовка. По умолчанию — `only-text`.

Возможные значения

`only-text`		Иконка не отображается. Отображается только заголовок кнопки.
`left-icon`		Иконка отображается слева от заголовка кнопки.
`right-icon`		Иконка отображается справа от заголовка кнопки.
`only-icon`		Отображается только иконка. Заголовок кнопки не отображается.

`caption` `string`

Локализуемый заголовок кнопки, который отображается при наведении курсора.

`icon` `string`

Иконка, которая отображается рядом с заголовком кнопки.

Возможные значения

`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	

Компонент `Menu-item`

`Menu-item` — компонент, который позволяет добавить дополнительные действия для пункта меню кнопки. Компонент `Menu-item` используется для любого взаимодействия, которое выполняет действие на текущей странице.

Свойства

`caption` `string`

Локализуемый заголовок пункта меню кнопки.

`visible` `boolean`

Признак, который управляет отображением пункта в выпадающем меню кнопки.

`disabled` `boolean`

Признак, который управляет блокировкой пункта в выпадающем меню кнопки. По умолчанию — `false` .

`items` `CrtMenuItemViewElementConfig[]`

Перечень пунктов вложенного меню кнопки. Отображается при наведении курсора на пункт меню, который содержит вложенное меню. Логика работы и свойства пунктов вложенного меню кнопки и самой кнопки аналогичны.

`icon` `string`

Иконка, которая отображается рядом с заголовком пункта меню.

Возможные значения

`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`

Цвет иконки в HEX-формате.

Компонент Slider JS

Основы

`Slider` — компонент, который позволяет настроить слайдер на странице приложения. Компонент `Slider` использует собственный элемент `<input>`, чтобы обеспечить доступность по умолчанию.

Свойства

`id` `string`

Уникальный идентификатор. Служебное поле.

`control` `FormControl`

Элемент управления.

`color` `string`

Стиль слайдера.

Возможные значения

`default`	Основной. Слайдер синего цвета.
`white`	Слайдер белого цвета на синем фоне.
`accent`	Слайдер зеленого цвета.
`primary`	Слайдер синего цвета.
`warn`	Слайдер красного цвета.

`ariaLabel` `string`

Свойство, которое связано с атрибутом `aria-label` элемента.

`value` `number`

Установленное значение слайдера. По умолчанию — `0`.

`minValue` `number`

Минимальное значение слайдера. По умолчанию — `0`.

`maxValue` `number`

Максимальное значение слайдера. По умолчанию — `100`.

`step` `number`

Значение шага слайдера. По умолчанию — `1`.

`disabled` `boolean`

Признак, который управляет доступностью изменения слайдера. По умолчанию — `false`.

`hideThumb` `boolean`

Признак, который скрывает ползунок слайдера. По умолчанию — `false`.

События

`valueChanged` `number`

Событие, которое генерируется при изменении значения слайдера.