Продукт Platform V UI Kits (UIK)

Компонент Библиотека визуальных компонентов UFSUI (UIUF)

Руководство по эксплуатации

Содержание

Руководство по эксплуатации компонента Библиотека визуальных компонентов UFSUI (UIUF)

Руководство прикладного разработчика компонента Библиотека визуальных компонентов UFSUI (UIUF)

Список терминов и определений, используемых в данном руководстве, приведен в одноименном разделе документа «Описание функциональных характеристик».

Системные требования

Системные требования к компоненту подробно описаны в документе «Руководство по установке», раздел «Системные требования».

Подключение и конфигурирование

Подключение библиотеки представляет собой скачивание и установку npm-пакетов с UI-компонентами. Подробная информация представлена в документе «Руководство по установке».

Миграция на текущую версию

Миграция на текущую версию происходит путем распаковки zip-архива соответствующей версии.

Быстрый старт

Использование библиотеки возможно после того, как все необходимые компоненты будут скачаны и установлены в прикладной проект, как указано в документе «Руководство по установке».

Использование программного компонента

В данном разделе описывается назначение и работа компонентов библиотеки.

Accordion

Компонент Аккордеон представляет собой вертикально сложенный список элементов. Каждый элемент может быть «развернут» или «раскрыт», чтобы показать содержимое, связанное с этим элементом.

AccordionItem

Для управления отображением каждого элемента используется компонент `AccordionItem`. Для корректной работы все элементы оборачиваются в компонент `Accordion`. Далее идет таблица свойства для компонента `AccordionItem`.

import

```
import { Accordion, AccordionItem } from '@v-uik/accordion'
```

Обычный аккордеон (без автоматического закрытия остальных элементов)

Аккордеон с одним открытым элементом

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Badge } from '@v-uik/badge'
import { Button } from '@v-uik/button'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { BadgeToggle } from '@v-uik/badge/examples'
import RawBadgeToggle from '!!raw-loader!@v-uik/badge/examples/BadgeToggle'

<Meta title={createTitle([COMPONENTS.feedback, 'Badge'])} component={Badge} /
>

export const children = (
  <div
    style={{
      backgroundColor: 'lightgrey',
      width: 40,
      height: 40,
    }}
  />
)

<Story
  name="Badge"
    args={{
    children,
    content: 1,
  }}
  argTypes={{
    children: {
      description: 'Содержимое компонента',
      control: false,
    },
  }}
>
  {(args) => <Badge {...args} />}
</Story>
```

Badge

Badge генерирует маленький значок в углу своего дочернего элемента.

import

```
import { Badge } from '@v-uik/badge'
```

Автономный значок без дочерних компонентов (Standalone)

Используется для отображения контента в одном из доступных цветовых состояний, например для отображения информации о новых уведомлениях.

```
<Badge content={1} />
<Badge content={1} status="error" />
<Badge content={1} status="info" />
<Badge content={1} status="neutral" />
<Badge content={1} status="warning" />
<Badge content={1} status="success" />
<Badge disabled content={1} />
```

Простые значки для компонента кнопки

На примере кнопки (целевой элемент, к которому применяется значок, может быть любым), рассмотрим отображение значков, с использованием разных цветов.

```
<Badge content={10}>
  <Button>Уведомления</Button>
</Badge>
<Badge content={10} status="success">
  <Button variant="outlined">Уведомления</Button>
</Badge>
<Badge content={10} status="error">
  <Button variant="ghost">Уведомления</Button>
</Badge>
<Badge content={10} status="info">
  <Button>Уведомления</Button>
</Badge>
<Badge content={10} status="warning">
  <Button variant="outlined">Уведомления</Button>
</Badge>
<Badge disabled content={10}>
  <Button variant="ghost" disabled>
    Уведомления
  </Button>
</Badge>
```

Превышение максимального значения

При превышении максимального значения бэйджа отрисовывается вспомогательный символ +

```
<Badge content={10} max={9}>
  <Button>Уведомления</Button>
</Badge>
```

Видимость значка

Когда свойство `content` имеет пустое значение — *0, undefined, null, '', false* — значок исчезает.

Если вы хотите, чтобы значок отображался всегда, можно использовать свойство `showZero`.

```
<Badge content={0} showZero>
  <div
    style={{
      height: '40px',
      width: '40px',
      backgroundColor: 'bisque',
    }}
  ></div>
</Badge>
```

Значок-точка

Свойство `dot` превращает значок в точку. Это можно использовать как уведомление о том, что что-то изменилось без количества.

Обратите внимание, свойство `dot` не отображает значок с пустым контентом. Убедитесь, что `content` не является пустым значением или указано свойство `showZero`

```
<Badge content={1} dot>
  <div
    style={{
      height: '40px',
      width: '40px',
      backgroundColor: 'bisque',
    }}
  ></div>
</Badge>
<Badge dot showZero>
  <div
    style={{
      height: '40px',
      width: '40px',
      backgroundColor: 'bisque',
    }}
  ></div>
</Badge>
```

Выравнивание значка

Когда значок отображается относительно дочернего элемента, вы можете управлять его позицией, с помощью свойства `position`, а также производить смещение по

вертикале и горизонтали, с помощью свойств `horizontalOffset` и `verticalOffset` соответственно.

```
  <Badge content={1} position="top-right">
    <div
      style={{
        height: '40px',
        width: '40px',
        backgroundColor: 'bisque',
      }}
    ></div>
  </Badge>
  <Badge content={1} position="bottom-right">
    <div
      style={{
        height: '40px',
        width: '40px',
        backgroundColor: 'bisque',
      }}
    ></div>
  </Badge>
  <Badge content={1} position="top-left">
    <div
      style={{
        height: '40px',
        width: '40px',
        backgroundColor: 'bisque',
      }}
    ></div>
  </Badge>
  <Badge content={1} position="bottom-left">
    <div
      style={{
        height: '40px',
        width: '40px',
        backgroundColor: 'bisque',
      }}
    ></div>
  </Badge>
```

```
import { Meta, Story, Canvas, ArgsTable, Source } from '@storybook/addon-docs
'

import {
  Bar,
  BarButton,
  BarMenuItem,
  BarDate,
  BarSearch,
  BarSelect,
  BarDropdown,
```

```
  BarDropdownItem,
  BarDivider,
  SubBar,
} from '@v-uik/bar'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { IconBurger } from '@v-uik/bar/examples/assets/IconBurger'
import { LogoSber } from '@v-uik/bar/examples/assets/LogoSber'
import {
  Basic,
  ContainerMock,
  BarMenuItemExample,
  BarSelectExample,
  BarAndSubBarExample,
  NestedMenuExample,
  BarHistoryExample,
} from '@v-uik/bar/examples'
import RawBarMenuItemExample from '!!raw-loader!@v-uik/bar/examples/BarMenuIt
emExample'
import RawBarSelectExample from '!!raw-loader!@v-uik/bar/examples/BarSelectEx
ample'
import RawBarAndSubBarExample from '!!raw-loader!@v-uik/bar/examples/BarAndSu
bBarExample'
import RawNestedMenuExample from '!!raw-loader!@v-uik/bar/examples/NestedMenu
Example'
import RawBarHistoryExample from '!!raw-loader!@v-uik/bar/examples/BarHistory
Example'
import RawCustomBarMenuItem from '!!raw-loader!@v-uik/bar/examples/CustomBarM
enuItem'

<Meta title={createTitle([COMPONENTS.navigation, 'Bar'])} component={Bar} />

<Story
  name="Bar"
>
  {(args) => <Basic {...args} />}
</Story>
```

Bar и Bar

Панель, которая располагается сверху или слева экрана и используется для быстрой навигации по основным разделам вашего приложения, отображения поля поиска, кнопок быстрый действий пользователя.

SubBar

Вспомогательная панель для расположения дополнительных компонентов. В SubBar могут использоваться все компоненты Bar*

import

```
import {
  Bar,
  BarMenuItem,
  BarDropdown,
  BarDropdownItem,
  BarSearch,
  BarDate,
  BarButton,
  BarDivider,
  SubBar,
} from '@v-uik/bar'
```

BarButton — быстрые действия пользователя

Для быстрых действий пользователя, например изменения состояния боковой панели, вы можете использовать компонент `BarButton`. Компонент поддерживает отображение иконки с помощью свойства `icon`.

```
<Bar>
  <BarButton icon={<IconBurger />} />
</Bar>
```

Навигации по приложению

Компонент в основном используется для отображения ссылок, которые используются для быстрой навигации по вашему приложению.

BarMenuItem API

BarDropdown API

BarDropdownItem API

Логотип

Для продвижения бренда продукта или быстрого перехода на главную страницу, вы можете добавить логотип на панель.

```
<Bar>
  <BarButton icon={<IconBurger />} />
  <LogoSber
    style={{
      margin: '12px 16px',
      fill: 'white',
    }}
  />
</Bar>
```

Поиск

Один из способов быстро найти необходимый контент на страницах приложения, это глобальный поиск. Компонент `BarSearch` предоставляет весь набор API компонента `Input`, но уже стилизованного для панели. Мы не реализуем алгоритмы поиска, так как это выходят за рамки функциональности библиотеки.

BarSearch API

```
<Bar>
  <BarButton icon={<IconBurger />} />
  <BarSearch style={{ marginLeft: 'auto' }} />
</Bar>
```

Меню выбора

Для переключения настроек из выбора возможных опций, которые влияют на отображение страницы, например изменение языка интерфейса.

BarSelect API

Системное время

Компонент для отображения текущей системной даты и времени.

BarDate API

```
<Bar>
  <BarButton icon={<IconBurger />} />
  <BarDate style={{ marginLeft: 'auto' }} />
</Bar>
```

Визуальный разделитель

Компонент-разделитель частей бара.

BarDivider API

```
<Bar>
  <BarButton icon={<IconBurger />} />
  <BarDivider />
  <BarDate />
</Bar>
```

Кастомизация компонентов

Пример кастомизации компонента для отображения иерархического меню. Данный пример может быть видоизменен, здесь он используется, как пример. Можно создать компонент принимающий массив детей, рекурсивно вызывающий себя, или использовать любой другой подход.

Иерархическое меню

Для создания иерархической вложенности, использован кастомный компонент. В примере показан вариант переноса текста, помимо переноса, можно увеличить ширину бара.

История переходов

Пример создания истории просмотра страниц с использованием кастомного компонента.

Возможные комбинации расположений и стилей Bar и SubBar

Вы можете комбинировать горизонтальное и вертикальное положения Bar и SubBar с помощью соответствующих свойств и добавления отступов.

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Box } from '@v-uik/box'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Heading, HeadingWithRef } from '@v-uik/box/examples'
import RawHeading from '!!raw-loader!@v-uik/box/examples/AsHeadingStory'
import RawHeadingWithRef from '!!raw-loader!@v-uik/box/examples/AsHeadingWith
Ref'

<Meta title={createTitle([COMPONENTS.utility, 'Box'])} component={Box} />

<Story
  name="Box"
  parameters={{
    docs: {
      disable: true,
    },
  }}
>
  {(...args) => <Box {...args} />}
</Story>
```

Box

Решает проблему типизации компонентов, которые динамически отображают HTML-тег компонента.

import

```
import { Box } from '@v-uik/box'
import type {
  PolymorphicComponentProps,
  PolymorphicComponent,
} from '@v-uik/box'
```

Собственный полиморфный компонент на основе компонента Box

Типы-утилиты для создания собственных полиморфных компонентов на основе компонента Box.

```
<Heading variant="h1">Заголовок 1 уровня</Heading>
```

Собственный полиморфный компонент на основе компонента Box с пробросом ref

Если нужно использовать проброс `ref` к вашему полиморфному компоненту, то это можно сделать так —

```
<HeadingWithRef variant="h1">
  Заголовок 1 уровня с пробросом ref
</HeadingWithRef>
```

Полезные статьи по теме

- NPM react-polymorphic-box
- Статья про PRC
- Статья про PRC with forwardRef

```
import { Meta, Story, ArgsTable, Canvas } from '@storybook/addon-docs'
import { Breadcrumbs } from '@v-uik/breadcrumbs'
import { Link } from '@v-uik/link'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Icon } from '@v-uik/breadcrumbs/examples/assets/Icon'

<Meta
  title={createTitle([COMPONENTS.navigation, 'Breadcrumbs'])}
  component={Breadcrumbs}
/>

<Story
  name="Breadcrumbs"
  argTypes={{
    children: {
      table: {
        disable: true,
      },
    },
  }}
  args={{
    items: [
      {
        children: 'Breadcrumb 1',
      },
      {
        children: 'Breadcrumb 2',
      },
      {
        disabled: true,
```

```
        children: 'Breadcrumb 3',
      },
      {
        children: 'Breadcrumb 4',
      },
      {
        children: 'Breadcrumb 5',
      },
      {
        children: 'Breadcrumb 6',
      },
      {
        children: 'Breadcrumb 7',
      },
      {
        children: 'Breadcrumb 8',
      },
      {
        children: 'Breadcrumb 9',
      },
      {
        children: 'Breadcrumb 10',
      },
    ],
  }}
>
  {({ items, ...args }) => (
    <Breadcrumbs {...args}>
      {items.map((item, index) => {
        if (index === items.length - 1) {
          return <span key={`${index}_${item.children}`}>{item.children}</spa
n>
        }
        return (
          <Link key={`${index}_${item.children}`} href="#" {...item}>
            {item.children}
          </Link>
        )
      })}
    </Breadcrumbs>
  )}
</Story>
```

Breadcrumbs

Показывает текущее расположение страницы в иерархии вашего приложении.

import

```
import { Breadcrumbs } from '@v-uik/breadcrumbs'
```

В основе хлебных крошек лежит компонент Link, который позволяет пользователю быстро вернуться в предыдущие разделы. Для отображения текущего раздела (последний элемент хлебных крошек) используйте обычный HTML элемент (span и т.п.)

```
  <Breadcrumbs>
    <Link href="#">Breadcrumb</Link>
    <Link href="#">
      <Icon style={{ marginRight: 8 }} />
      Breadcrumb
    </Link>
    <Link disabled href="#">
      <Icon style={{ marginRight: 8 }} />
      Breadcrumb disabled
    </Link>
    <span>Breadcrumb</span>
  </Breadcrumbs>
```

Скрытие элементов в случае длинного списка

```
  <Breadcrumbs maxItems={3}>
    <Link href="#">Breadcrumb 1</Link>
    <Link href="#">Breadcrumb 2</Link>
    <Link href="#">Breadcrumb 3</Link>
    <Link href="#">Breadcrumb 4</Link>
    <span>Breadcrumb 5</span>
  </Breadcrumbs>
```

Полезные статьи по теме

- Breadcrumb Navigation Increasingly Useful

Связанные компоненты

- Link

```
import { Meta, Story, Canvas, ArgsTable } from '@storybook/addon-docs'
import { Button } from '@v-uik/button'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Icon } from '@v-uik/button/examples/assets/Icon'

<Meta title={createTitle([COMPONENTS.controls, 'Button'])} component={Button}
/>

<Story
  name="Button"
  args={{ children: 'Button' }}
>
  {(args) => <Button {...args} />}
</Story>
```

Button

Кнопки позволяют пользователям выполнять действия и делать выбор одним нажатием.

import
```
import { Button } from '@v-uik/button'
```

Заполненные кнопки (default)

Данный тип кнопок имеет высокий акцент.

```
  <Button>Primary</Button>
  <Button color="secondary">Secondary</Button>
  <Button color="error">Error</Button>
```

Контурные кнопки
```
  <Button kind="outlined">Primary</Button>
  <Button kind="outlined" color="secondary">
    Secondary
  </Button>
  <Button kind="outlined" color="error">
    Error
  </Button>
```

Текстовые кнопки
```
  <Button kind="ghost">Primary</Button>
  <Button kind="ghost" color="secondary">
    Secondary
  </Button>
  <Button kind="ghost" color="error">
    Error
  </Button>
```

Кнопки с иконками
```
  <Button style={{ minWidth: 40, padding: 7 }}>
    <Icon />
  </Button>
  <Button>
    <Icon style={{ marginRight: 8 }} />
    Text after icon
  </Button>
  <Button color="secondary">
    Text before icon
    <Icon style={{ marginLeft: 8 }} />
  </Button>
```

Кнопки разных размеров
```
  <Button size="sm">
    <Icon width={16} height={16} style={{ marginRight: 8 }} />
    Small
```

```
  </Button>
  <Button>
    <Icon style={{ marginRight: 8 }} />
    Medium (default)
  </Button>
  <Button size="lg">
    <Icon style={{ marginRight: 8 }} />
    Large
  </Button>

import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { ButtonGroup } from '@v-uik/button-group'
import { Button } from '@v-uik/button'
import { createTitle, COMPONENTS, DocsAlert } from '../../docs/showroom/confi
g'
import {
  Single,
  Multi,
  Examples,
  PopupExample,
} from '@v-uik/button-group/examples'
import RawSingle from '!!raw-loader!@v-uik/button-group/examples/Single'
import RawMulti from '!!raw-loader!@v-uik/button-group/examples/Multi'
import RawExamples from '!!raw-loader!@v-uik/button-group/examples/Examples'
import RawPopupExample from '!!raw-loader!@v-uik/button-group/examples/PopupE
xample'

<Meta
  title={createTitle([COMPONENTS.controls, 'ButtonGroup'])}
  component={ButtonGroup}
/>

<Story
  name="ButtonGroup"
  argTypes={{
    children: {
      table: {
        disable: true,
      },
    },
  }}
  args={{
    items: [
      {
        name: 'first',
        children: 'button 1',
      },
      {
        name: 'second',
```

```
        disabled: true,
        children: 'button 2',
      },
      {
        name: 'third',
        children: 'long button 3',
      },
      {
        name: 'fourth',
        children: 'button 4',
      },
    ],
    value: 'first',
  }}
>
  {({ items, ...args }) => (
    <ButtonGroup {...args}>
      {items.map((item, index) => (
        <Button key={`${index}_${item.name}`} {...item} />
      ))}
    </ButtonGroup>
  )}
</Story>
```

ButtonGroup

Отображает группу связанных кнопок.

import

```
import { ButtonGroup } from '@v-uik/button-group'
import { Button } from '@v-uik/button'
```

Размеры и цветовая схема

Единичный выбор

- с возможность выбора одного значения:

С возможность выбора нескольких значений

Отображение Popup внутри ButtonGroup

Компонент ButtonGroup советуется использовать только для одной цели - отображение кнопок в ряд и получения состояния selected. Компонент ButtonGroup позволяет использовать внутри себя только компонент Button, не обернутый в обертки (любой тег или компонент)

Связанные компоненты

- Button

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
```

```
import { Checkbox } from '@v-uik/checkbox'
import { CheckboxGroup } from '@v-uik/checkbox-group'
import { LabelControl } from '@v-uik/label-control'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'

import {
  WithLabel,
  Disabled,
  Indeterminate,
  LabelPosition,
  CheckboxGroupVertical,
} from '@v-uik/checkbox/examples'
import RawWithLabel from '!!raw-loader!@v-uik/checkbox/examples/WithLabel'
import RawDisabled from '!!raw-loader!@v-uik/checkbox/examples/Disabled'
import RawIndeterminate from '!!raw-loader!@v-uik/checkbox/examples/Indetermi
nate'
import RawLabelPosition from '!!raw-loader!@v-uik/checkbox/examples/LabelPosi
tion'
import RawCheckboxGroupVertical from '!!raw-loader!@v-uik/checkbox/examples/C
heckboxGroupVertical'

<Meta
  title={createTitle([COMPONENTS.controls, 'Checkbox'])}
  component={Checkbox}
/>

<Story
  name="Checkbox"
  args={{
    children: 'Checkbox',
  }}
  argTypes={{
    children: {
      description: 'Содержимое компонента',
      control: {
        type: 'text',
      },
    },
  }}
>
  {(args) => <Checkbox {...args} />}
</Story>
```

Checkbox

Чекбокс используется для включения или выключения нескольких опций.

Checkbox API

CheckboxGroup API

import

```
import { Checkbox } from '@v-uik/checkbox'
import { CheckboxGroup } from '@v-uik/checkbox-group'
```

Чекбокс с ярлыком

Вертикальный список флажков

Неопределенное состояние

Эта опция включает режим отображения, при котором не все элементы внутри группы выбраны.

Местоположение метки относительно чекбокса

Состояние disabled

В этом состоянии пользователь не может взаимодействовать с чекбоксом.

Связанные компоненты

- LabelControl

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { ComboBox } from '@v-uik/combo-box'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'

import {
  CompactTags,
  DividerOptions,
  Error,
  Grouping,
  MultipleSelect,
  OptionsWithIcon,
  OptionsWithTitle,
  OutsideTags,
  Searchable,
  SingleSelect,
} from './examples'

import RawCompactTags from '!!raw-loader!./examples/CompactTags'
import RawDividerOptions from '!!raw-loader!./examples/DividerOptions'
import RawError from '!!raw-loader!./examples/Error'
import RawMultipleSelect from '!!raw-loader!./examples/MultipleSelect'
import RawOptionsWithIcon from '!!raw-loader!./examples/OptionsWithIcon'
import RawOptionsWithTitle from '!!raw-loader!./examples/OptionsWithTitle'
import RawOutsideTags from '!!raw-loader!./examples/OutsideTags'
```

```
import RawSingleSelect from '!!raw-loader!./examples/SingleSelect'
import RawSearchable from '!!raw-loader!./examples/Searchable'
import RawGrouping from '!!raw-loader!./examples/Grouping'

<Meta
  title={createTitle([COMPONENTS.inputFields, 'ComboBox'])}
  component={ComboBox}
/>

export const options = [
  { value: '1', label: 'Опция 1' },
  { value: '2', label: 'Опция 2' },
  { value: '3', label: 'Опция 3' },
  { value: '4', label: 'Опция 4' },
  { value: '5', label: 'Опция 5' },
  { value: '6', label: 'Длинная опция 6' },
  { value: '7', label: 'Опция 7' },
  { value: '8', label: 'Опция 8' },
  { value: '9', label: 'Опция 9' },
  { value: '10', label: 'Опция 10' },
  { value: '11', label: 'Опция 11' },
  { value: '12', label: 'Длинная опция 12' },
  { value: '13', label: 'Опция 13' },
  { value: '14', label: 'Опция 14' },
  { value: '15', label: 'Опция 15' },
]

<Story
  name="ComboBox"
  parameters={{
    docs: {
      disable: true,
    },
  }}
  args={{
    value: '',
    options: options,
  }}
  decorators={[
    (Story) => (
      <div style={{ height: '150vh' }}>
        <Story />
      </div>
    ),
  ]}
>
  {(args) => <ComboBox {...args} />}
</Story>
```

ComboBox

Усложненный вариант базового селекта, имеет в себе функционал фильтрации, мультивыбора, очистки поля. Одиночный выбор рекомендуется использовать, когда нужна фильтрация по полю. В остальных случаях рекомендуется использовать Select

import

```
import { ComboBox } from '@v-uik/combo-box'
```

Единичный выбор

За открытие выпадающего списка при фокусе на элементе отвечает свойство `openOnFocus`.

Множественный выбор

Для множественного выбора, используется параметр `multiple`. За открытие выпадающего списка при фокусе на элементе отвечает свойство `openOnFocus`.

Чтобы выпадающий список не закрывался при выборе нескольких значений, используйте свойство `disableCloseOnSelect`. В этом случае список закрывается по клику за пределами вы списка.

Множественный выбор с поиском

Ошибка валидации

Отображение опций за пределом комбо бокса

В кейсах, когда выбранные значения нужно отобразить за пределами поля выбора, можно воспользоваться параметром `disableVisibleData` для того, чтобы данные не отображались внутри поля. После чего отобразить value в любом удобном формате, например в тегах.

Разделитель между опциями

Разделители опций можно настроить через объект `listProps`, параметром `stripe`.

Замена содержимого опций

Внутри опций выпадающего списка можно размещать кастомное содержимое. За это поведение отвечают свойства `prefix` и `suffix` массива с опциями.

Опции с заголовком и описанием

Обычно для отображения значения мы используем свойство `label`, которое можно считать заголовком опции. Если вам требуется добавить описание, вы можете использовать свойство `description`. В этом случаев `label` получит жирное начертание, а `description` отобразится под `label`.

Компактное отображение тегов

Для компактных интерфейсов существует решение по отображению n-тегов, или в случае 0, просто отображается счетчик. За это отвечает параметр `limitTags`. Если указать отрицательное значение, будет считаться как 0

Группировка

Группирует элементы с помощью опции `groupBy`.

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import {
  DatePicker,
  DateLibAdapterProvider,
  CalendarPicker,
  DayView,
  MonthView,
  YearView,
  RangeDayView,
} from '@v-uik/date-picker'
import { createTitle, COMPONENTS, DocsAlert } from '../../docs/showroom/confi
g'

import {
  BasicStory,
  DayViewStory,
  MonthViewStory,
  YearViewStory,
  CalendarPickerStory,
  RenderInputStory,
  RangeDayViewStory,
} from './examples'

import RawBasic from '!!raw-loader!./examples/BasicStory'
import RawDayViewStory from '!!raw-loader!./examples/DayViewStory'
import RawMonthViewStory from '!!raw-loader!./examples/MonthViewStory'
import RawYearViewStory from '!!raw-loader!./examples/YearViewStory'
import RawCalendarPickerStory from '!!raw-loader!./examples/CalendarPickerSto
ry'
import RawRenderInputStory from '!!raw-loader!./examples/RenderInputStory'
import RawRangeDayViewStory from '!!raw-loader!./examples/RangeDayViewStory'

<Meta
  title={createTitle([COMPONENTS.inputFields, 'DatePicker'])}
  component={DatePicker}
/>

<Story
  name="DatePicker"
```

```
>
  {(args) => <DatePicker {...args} />}
</Story>
```

DatePicker

Компонент выбора даты.

import

```
import { DatePicker, DateLibAdapterProvider } from '@v-uik/date-picker'
```

DatePicker API

DateLibAdapterProvider API

Перед началом работы

Для парсинга дат, форматирование и тому подобного, мы используем адаптер date-io, который позволяет использовать ту же библиотеку дат, которая у вас уже установлена в приложении

Чтобы подключить библиотеку дат, нам нужно использовать один из существующих адаптеров или написать свой.

```
import { DateFnsAdapter } from '@v-uik/date-picker/dist/adapters/date-fns'
import { DateFnsJalaliAdapter } from '@v-uik/date-picker/dist/adapters/date-fns-jalali'
import { DayjsAdapter } from '@v-uik/date-picker/dist/adapters/dayjs'
import { HijriAdapter } from '@v-uik/date-picker/dist/adapters/hijri'
import { JalaaliAdapter } from '@v-uik/date-picker/dist/adapters/jalaali'
import { LuxonAdapter } from '@v-uik/date-picker/dist/adapters/luxon'
import { MomentAdapter } from '@v-uik/date-picker/dist/adapters/moment'
```

С помощью компонента `DateLibAdapterProvider`, мы пробросим необходимое API для работы с датами для всех компонентов `DatePicker`. Подключить этот компонент нужно один раз, в файле инициализации вашего приложения.

```
import * as React from 'react'
import { DatePicker, DateLibAdapterProvider } from '@v-uik/date-picker'
import { DateFnsAdapter } from '@v-uik/date-picker/adapters/date-fns'
import { ru } from 'date-fns/locale'

export function App() {
  return (
    <DateLibAdapterProvider
      dateAdapter={DateFnsAdapter}
      options={{ locale: ru }}
    >
      <App />
    </DateLibAdapterProvider>
```

```
  )
}
```

Базовый пример — выбора времени

Замена поля ввода времени

Благодаря рендер свойству `renderInput` вы можете отобразить любой компонент. В нашем примере, вместо поля ввода, мы отобразим кнопку, которая вызывает календарь.

Саб-компоненты

Некоторые низкоуровневые компоненты, которые используется в пакете `@v-uik/date-picker`. Вы можете использовать их для создания собственных компонентов работы с датами.

Саб-компонент DayView

Используется для отображения календаря.

DayView API

Саб-компонент RangeDayView

Используется для отображения календаря с диапазоном выбора дат.

RangeDayView API

Саб-компонент MonthView

Используется для отображения месяцев.

MonthView API

Саб-компонент YearView

Используется для отображения лет.

YearView API

Саб-компонент CalendarPicker

Обертка для выбора даты, которая объединяет выбор даты, месяца и года.

CalendarPicker API

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { RangePicker } from '@v-uik/date-picker'
```

```
import { createTitle, COMPONENTS, DocsAlert } from '../../docs/showroom/confi
g'
import { RestrictedDays, RenderInput } from './examples'

import RawRestrictedDays from '!!raw-loader!./examples/RestrictedDays'
import RawRenderInput from '!!raw-loader!./examples/RenderInput'

<Meta
  title={createTitle([COMPONENTS.inputFields, 'RangePicker'])}
  component={RangePicker}
/>

<Story
  name="RangePicker"
>
  {(args) => <RangePicker {...args} />}
</Story>
```

RangePicker

Компонент выбора диапазона дат.

Для работы компонента, его необходимо обернуть в DateLibAdapterProvider (можно обернуть все приложение для всех пикеров в нем), которому необходимо передать в свойство dateAdapter экземпляр адаптера библиотеки по работе с датами, находящегося в @v-uik/date-picker/adapters (или же самописный адаптер, имплементирующий его интерфейс). Таким образом, компонент не зависит от конкретной библиотеки обработки дат.

import

```
import { RangePicker } from '@v-uik/date-picker'
```

Пример, показывающий возможности ограничения выбора дат

в примере заблокированы даты будущего и выходные.

Пример пикера на кастомном компоненте

Вы можете использовать пикер на любых компонентах с помощью свойства `renderInput`. В случае использования кастомных инпутов, не забудьте пробросить свойства `inputRef`, `inputProps`, `value` и `onChange`.

Связанные компоненты

- Dropdown

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Drawer, DrawerHeader, DrawerFooter } from '@v-uik/drawer'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import {
```

```
  BasicDrawer,
  PositionsExample,
  NonModalExample,
  ContainerExample,
  ContainerOverflowExample,
  MultiLevelExample,
} from '@v-uik/drawer/examples'
import RawBasicDrawer from '!!raw-loader!@v-uik/drawer/examples/BasicDrawer'
import RawPositionsExample from '!!raw-loader!@v-uik/drawer/examples/Position
sExample'
import RawNonModalExample from '!!raw-loader!@v-uik/drawer/examples/NonModalE
xample'
import RawContainerExample from '!!raw-loader!@v-uik/drawer/examples/Containe
rExample'
import RawContainerOverflowExample from '!!raw-loader!@v-uik/drawer/examples/
ContainerOverflowExample'
import RawMultiLevelExample from '!!raw-loader!@v-uik/drawer/examples/MultiLe
velExample'

<Meta
  title={createTitle([COMPONENTS.navigation, 'Drawer'])}
  component={Drawer}
/>

<Story
  name="Drawer"
>
  {(args) => <Drawer {...args} />}
</Story>
```

Drawer

Боковая панель для предоставления какой-либо информации, не требующей постоянного отображения на странице.

import

```
import { Drawer, DrawerHeader, DrawerBody, DrawerFooter } from '@v-uik/drawer
'
```

DrawerHeader

Компонент DrawerHeader используется для отображения заголовка боковой панели.

DrawerBody

Компонент DrawerBody используется для отображения содержимого боковой панели.

DrawerFooter

Компонент DrawerFooter используется для отображения футера боковой панели.

Базовый пример

Варианты расположения на странице

С помощью свойства `position` вы можете выбрать с какой стороны экрана будет появляться панель.

Пример без блокировки страницы

Установив свойству `backdrop` значение `false` можно отключить затемнение фона, также указав свойство `bodyScrollLock = false` отключается блокировка скролла страницы. Заметьте, что компонент будет отрисован в текущее место в DOM-дереве, тогда как вариант по-умолчанию рендерит панель в конец документа.

Панель для пользовательского элемента

С помощью свойства `container` можно отобразить боковую панель внутри любого элемента DOM-дерева.

Панель для пользовательского элемента со скроллом

Для того чтобы использовать Drawer на скроллящемся элементе, рекомендуется обернуть его каким-либо элементом-контейнером, и уже его передавать свойству `container`.

Многоуровневая панель

Немного стилизовав компоненты Drawer, можно добиться эффекта вложенности панелей.

```
import { Meta, Story, Canvas, ArgsTable } from '@storybook/addon-docs'
import { Dropdown } from '@v-uik/dropdown'
import { Button } from '@v-uik/button'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'

<Meta
  title={createTitle([COMPONENTS.utility, 'Dropdown'])}
  component={Dropdown}
/>

export const content = (
  <div style={{ padding: 10, backgroundColor: '#ccc', borderRadius: 4 }}>
    Dropdown content
  </div>
)

export const children = (
  <Button style={{ margin: '50px 0 0 200px' }}>click me</Button>
)
```

```
<Story
  name="Dropdown"
  parameters={{
    docs: {
      disable: true,
    },
  }}
  args={{
    children: children,
    content: content,
  }}
  argTypes={{
    children: {
      control: false,
    },
    content: {
      control: false,
    },
    open: {
      control: false,
    },
  }}
>
  {(args) => <Dropdown {...args} />}
</Story>
```

Dropdown

Компонент Dropdown является более функциональной оберткой Popup. Компонент-потомок Dropdown становится триггером и ориентиром отображения Popup по действию переданному в свойство action (hover, click, contextMenu). Содержимое Popup передается в свойство content.

import

```
import { Dropdown } from '@v-uik/dropdown'
```

Выпадающий компонент при на нажатии на кнопку

```
  <Dropdown
    content={
      <div style={{ padding: 10, backgroundColor: '#ccc', borderRadius: 4 }}>
        Dropdown content
      </div>
    }
  >
    <Button>Нажми меня</Button>
  </Dropdown>
```

Связанные компоненты

- Popup

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { DropdownMenu, DropdownMenuItem } from '@v-uik/dropdown-menu'
import { createTitle, COMPONENTS } from '@v-uik/showroom/config'
import { Cascading } from '@v-uik/dropdown-menu/examples'
import RawCascading from '!!raw-loader!@v-uik/dropdown-menu/examples/Cascadin
g'

<Meta
  title={createTitle([COMPONENTS.navigation, 'DropdownMenu'])}
  component={DropdownMenu}
/>

export const content = (
  <>
    <DropdownMenuItem>Option 1</DropdownMenuItem>
    <DropdownMenuItem>Option 2</DropdownMenuItem>
    <DropdownMenuItem disabled>Option 3</DropdownMenuItem>
    <DropdownMenuItem
      dropdownProps={{
        content: (
          <>
            <DropdownMenuItem>Option 1</DropdownMenuItem>
            <DropdownMenuItem>Option 2</DropdownMenuItem>
            <DropdownMenuItem>Option 3</DropdownMenuItem>
          </>
        ),
      }}
    >
      Option 4
    </DropdownMenuItem>
  </>
)

export const children = (
  <div
    style={{
      maxWidth: 400,
      backgroundColor: 'lightgrey',
      height: 32,
      width: 60,
      borderRadius: 4,
    }}
  />
)

<Story
  name="DropdownMenu"
```

```
  argTypes={{
    children: {
      control: false,
    },
    content: {
      control: false,
    },
  }}
  args={{
    children,
    content,
  }}
>
  {(args) => <DropdownMenu {...args} />}
</Story>
```

DropdownMenu

Выпадающее меню.

import

```
import { DropdownMenu, DropdownMenuItem } from '@v-uik/dropdown-menu'
```

DropdownMenu API

DropdownMenuItem API

Внутренний компонент выпадающего меню.

Связанные компоненты

- Dropdown

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Grid, GridItem } from '@v-uik/grid'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Adaptive, CurrentMedia } from '@v-uik/grid/examples'
import RawAdaptive from '!!raw-loader!@v-uik/grid/examples/Adaptive'

<Meta title={createTitle([COMPONENTS.utility, 'Grid'])} component={Grid} />

export const colors = ['#eee', 'coral', 'palevioletred', 'lightblue']

export const getCellStyle = (color) => ({
  backgroundColor: color,
  padding: 20,
  borderRadius: 4,
  textAlign: 'center',
  boxShadow: `0 2px 2px ${color}`,
})
```

```
<Story
  name="Grid"
  parameters={{
    docs: {
      disable: true,
    },
  }}
  args={{
    items: [
      {
        xs: 16,
        sm: 8,
        md: 6,
        lg: 4,
        xl: 3,
        xxl: 2,
        children: 'xs=16, sm=8, md=6, lg=4, xl=3, xxl=2',
      },
      {
        xs: 16,
        sm: 8,
        md: 6,
        lg: 4,
        xl: 3,
        xxl: 2,
        children: 'xs=16, sm=8, md=6, lg=4, xl=3, xxl=2',
      },
      {
        xs: 16,
        sm: 8,
        md: 6,
        lg: 4,
        xl: 3,
        xxl: 2,
        children: 'xs=16, sm=8, md=6, lg=4, xl=2, xxl=2',
      },
    ],
  }}
>
  {({ items, ...args }) => (
    <Grid {...args}>
      {items.map((item, index) => {
        const color =
          index === 0 || index === 1
            ? colors[index]
            : colors[index % colors.length]
        return (
          <GridItem key={`${index}_${item.children}`} {...item}>
            <div style={getCellStyle(color)}>{item.children}</div>
          </GridItem>
```

```
        )
      })}
    </Grid>
  )}
</Story>
```

Grid

Компонент-контейнер для построения разметочной сетки. Состоит из 16 колонок, основана на flex и работает с использованием media query.

import

```
import { Grid, GridItem } from '@v-uik/grid'
```

Grid API

GridItem API

Элемент сетки контейнера.

Адаптивная сетка

Размеры блоков меняются вместе с размерами вашего экрана и элементы занимают разное пространство.

```
import { Meta, Story, Canvas, ArgsTable } from '@storybook/addon-docs'
import { Input } from '@v-uik/input'
import { InputHelperText } from '@v-uik/input-helper-text'
import { InputLabel } from '@v-uik/input-label'
import { createTitle, COMPONENTS } from '@v-uik/showroom/config'
import { Icon } from '@v-uik/input/examples/assets/Icon'

<Meta
  title={createTitle([COMPONENTS.inputFields, 'Input'])}
  component={Input}
/>

<Story
  name="Input",
  }}
>
  {(args) => <Input {...args} />}
</Story>
```

Input

Компонент ввода текста.

import

```
import { Input } from '@v-uik/input'
import { InputLabel } from '@v-uik/input-label'
import { InputHelperText } from '@v-uik/input-helper-text'
```

Input API

InputLabel API

Это саб-компонент отображения лейбла для различных полей ввода, который можно настроить через свойство `labelProps`.

InputHelperText API

Это саб-компонент отображения вспомогательного текста для различных полей ввода, который можно настроить через свойство `helperTextProps`.

Поле ввода с ярлыком и описанием

```
  <Input label="Введите логин" helperText="Описание поля" />
```

Поле ввода с ярлыком и подсказкой

```
  <Input
    label="Введите логин"
    labelProps={{
      tooltipText: 'Подсказка',
      tooltipProps: {
        single: true,
        dropdownProps: {
          modifiers: [
            {
              name: 'offset',
              options: {
                offset: [-5, 8],
              },
            },
          ],
        },
      },
    }}
  />
```

Варианты индикации ошибки

```
  <Input
    error
    showErrorIcon={false}
    label="Введите логин"
    helperText="Поле не может быть пустым"
  />
  <Input error label="Введите логин" helperText="Поле не может быть пустым" /
>
```

```
  <Input
    error
    showErrorIcon
    errorIconTooltipProps={{
      dropdownProps: {
        placement: 'top',
        content: 'Поле не может быть пустым',
      },
    }}
    label="Введите логин"
  />
```

Поле ввода с иконками в начале и конце

```
  <Input prefix={<Icon />} suffix={<Icon />} />
```

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { InputNumber } from '@v-uik/input-number'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Example } from '@v-uik/input-number/examples/Example'
import RawExample from '!!raw-loader!@v-uik/input-number/examples/Example'

<Meta
  title={createTitle([COMPONENTS.inputFields, 'InputNumber'])}
  component={InputNumber}
/>

<Story
  name="InputNumber"
  args={{
    value: 12345.67,
  }}
>
  {(args) => <InputNumber {...args} />}
</Story>
```

InputNumber

Компонент поля ввода числовых значений

```
import { InputNumber } from '@v-uik/input-number'
```

Отображение дробного числа

Связанные компоненты

- Input

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { InputPassword } from '@v-uik/input-password'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
```

```
import { BasicInputPassword } from '@v-uik/input-password/examples/BasicInput
Password'
import RawBasicPasswordInput from '!!raw-loader!@v-uik/input-password/example
s/BasicInputPassword'
import { LeftIconExample } from '@v-uik/input-password/examples/LeftIconExamp
le'
import RawLeftIconExample from '!!raw-loader!@v-uik/input-password/examples/L
eftIconExample'
import { CustomIcon } from '@v-uik/input-password/examples/CustomIcon'
import RawCustomIcon from '!!raw-loader!@v-uik/input-password/examples/Custom
Icon'

<Meta
  title={createTitle([COMPONENTS.inputFields, 'InputPassword'])}
  component={InputPassword}
/>

<Story
  name="InputPassword"
  parameters={{
    docs: {
      disable: true,
    },
  }}
>
  {(args) => <InputPassword {...args} value="Top secret" />}
</Story>
```

InputPassword

Компонент ввода пароля.

import

```
import { InputPassword } from '@v-uik/input-password'
```

Пример

Иконка слева

Кастомная иконка

```
import { Meta, Story, ArgsTable, Canvas } from '@storybook/addon-docs'
import { LabelControl } from '@v-uik/label-control'
import { Checkbox } from '@v-uik/checkbox'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'

<Meta
  title={createTitle([COMPONENTS.utility, 'LabelControl'])}
  component={LabelControl}
/>
```

```
<Story
  name="LabelControl"
  parameters={{
    docs: {
      disable: true,
    },
  }}
  args={{
    label: 'Checkbox label',
  }}
>
  {(args) => <LabelControl {...args} control={<Checkbox />} />}
</Story>
```

LabelControl

Компонент для отображения Radio, CheckBox и Switch совместно с заголовком

import

```
import { LabelControl } from '@v-uik/label-control'
```

Связанные компоненты

- Switch
- Radio
- CheckBox

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Link } from '@v-uik/link'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { SimpleExample } from '@v-uik/link/examples/SimpleExample'
import RawSimpleExample from '!!raw-loader!@v-uik/link/examples/SimpleExample
'
import { Disabled } from '@v-uik/link/examples/Disabled'
import RawDisabled from '!!raw-loader!@v-uik/link/examples/Disabled'
import { UnderlineExample } from '@v-uik/link/examples/UnderlineExample'
import RawUnderlineExample from '!!raw-loader!@v-uik/link/examples/UnderlineE
xample'
import RawLinkWithComponent from '!!raw-loader!@v-uik/link/examples/LinkWithC
omponent'

<Meta title={createTitle([COMPONENTS.navigation, 'Link'])} component={Link} /
>

<Story
  name="Link"
  args={{
    children: 'link',
  }}
  argTypes={{
```

```
    children: {
      description: 'Содержимое компонента',
      control: {
        type: 'text',
      },
    },
  }}
>
  {(args) => <Link href="#" {...args} />}
</Story>
```

Link

Компонент Link используется для обозначения ссылок.

```
import { Link } from '@v-uik/link'
```

Ссылка в тексте

Ссылка с подчеркиванием

Отключенная ссылка

Примеры работы с библиотеками маршрутизации

Демонстрация работы компонента с библиотекой react-router. Для интеграции со сторонними библиотеками мы рекомендуем использовать параметр as.

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { List, ListItem, ListItemGroup } from '@v-uik/list'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'

import {
  Grouping,
  Interactive,
  MultilineDisplay,
  Sizes,
} from '@v-uik/list/examples'

import RawInteractive from '!!raw-loader!@v-uik/list/examples/Interactive'
import RawSizes from '!!raw-loader!@v-uik/list/examples/Sizes'
import RawMultilineDisplay from '!!raw-loader!@v-uik/list/examples/MultilineD
isplay'
import RawGrouping from '!!raw-loader!@v-uik/list/examples/Grouping'

<Meta title={createTitle([COMPONENTS.dataDisplay, 'List'])} component={List}
/>

export const items = [
  { children: 'первый элемент списка' },
```

```
  { children: 'второй элемент списка' },
  { children: 'третий элемент списка' },
]

<Story
  name="List"
  parameters={{
    docs: {
      disable: true,
    },
  }}
  args={{
    items,
  }}
  argTypes={{
    children: {
      table: {
        disable: true,
      },
    },
  }}
>
  {({ items, ...args }) => (
    <List {...args}>
      {items.map((item, index) => (
        <ListItem key={`item_${index}`} {...item} />
      ))}
    </List>
  )}
</Story>
```

List

Компонент List используется для отображения списков.

ListItem API

Компонент ListItem используется для отображения элементы списка.

ListItemGroup API

Группировка элементов списка. Заголовок группы устанавливается с помощью свойства `label`.

```
import { List, ListItem, ListItemGroup } from '@v-uik/list'
```

Пример с интерактивными элементами

Пример с многострочными опциями

Пример с разными размерами элементов

Группировка

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { MaskedInput } from '@v-uik/masked-input'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Example } from '@v-uik/masked-input/examples/Example'
import RawExample from '!!raw-loader!@v-uik/masked-input/examples/Example'

<Meta
  title={createTitle([COMPONENTS.inputFields, 'MaskedInput'])}
  component={MaskedInput}
/>

<Story
  name="MaskedInput"
  args={{
    mask: '+7 (111) 111-11-11',
    placeholder: '+7 (999) 123-45-67',
  }}
  argTypes={{
    formatCharacters: {
      control: {
        disable: true,
      },
    },
  }}
>
  {(args) => <MaskedInput {...args} />}
</Story>
```

MaskedInput

Компонент ввода значения по маске.

import

```
import { MaskedInput } from '@v-uik/masked-input'
```

FormatCharacters

Объект, ключами которого являются символы маски, а значение - объект с полями ‘validate’ (значением которого будет функция, принимающая параметр ‘char’ - проверяемый символ, и возвращающая булево значение, которое сообщает подходит ли символ по маске) и ‘transform’ (необязательное поле, значением которого должна

быть функция, принимающая параметр ‘char’ - проверяемый символ, и возвращающая преобразованный символ).

```
export const DIGIT_REGEXP = /^\d$/
export const LETTER_REGEXP = /^[A-Za-zA-Яа-яёЁ]$/
export const CYRILLIC_REGEXP = /^[А-Яа-яёЁ]$/
export const LATIN_REGEXP = /^[A-Za-z]$/
export const ALPHANUMERIC_REGEXP = /^[\dA-Za-zA-Яа-яёЁ]$/
export const DEFAULT_FORMAT_CHARACTERS = {
  '*': {
    validate(char) {
      return ALPHANUMERIC_REGEXP.test(char)
    },
  },
  '1': {
    validate(char) {
      return DIGIT_REGEXP.test(char)
    },
  },
  a: {
    validate(char) {
      return LETTER_REGEXP.test(char)
    },
  },
  A: {
    validate(char) {
      return LETTER_REGEXP.test(char)
    },
    transform(char) {
      return char.toUpperCase()
    },
  },
  я: {
    validate(char) {
      return CYRILLIC_REGEXP.test(char)
    },
  },
  Я: {
    validate(char) {
      return CYRILLIC_REGEXP.test(char)
    },
    transform(char) {
      return char.toUpperCase()
    },
  },
  l: {
    validate(char) {
      return LATIN_REGEXP.test(char)
    },
  },
```

```
  L: {
    validate(char) {
      return LATIN_REGEXP.test(char)
    },
    transform(char) {
      return char.toUpperCase()
    },
  },
  '#': {
    validate(char) {
      return ALPHANUMERIC_REGEXP.test(char)
    },
    transform(char) {
      return char.toUpperCase()
    },
  },
}
```

Валидация номера телефона для региона Россия (+7)

Связанные компоненты

- Input

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Modal, ModalHeader } from '@v-uik/modal'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Basic } from '@v-uik/modal/examples/Basic'
import RawBasic from '!!raw-loader!@v-uik/modal/examples/Basic'
import { CustomWidth } from '@v-uik/modal/examples/CustomWidth'
import RawCustomWidth from '!!raw-loader!@v-uik/modal/examples/CustomWidth'
import { BodyOverflow } from '@v-uik/modal/examples/BodyOverflow'
import RawBodyOverflow from '!!raw-loader!@v-uik/modal/examples/BodyOverflow'
import { DisabledHandle } from '@v-uik/modal/examples/DisabledHandle'
import RawDisabledHandle from '!!raw-loader!@v-uik/modal/examples/DisabledHan
dle'

<Meta title={createTitle([COMPONENTS.feedback, 'Modal'])} component={Modal} /
>

<Story
  name="Modal"
  argTypes={{
    open: {
      table: {
        disable: true,
      },
    },
    children: {
      table: {
```

```
        disable: true,
      },
    },
  }}
>
  {(args) => <Modal {...args} />}
</Story>
```

Modal

Компонент Modal используется для отображения модальных окон.

import

```
import { Modal, ModalHeader, ModalBody, ModalFooter } from '@v-uik/modal'
```

ModalHeader

Компонент ModalHeader используется для отображения заголовка модального окна.

ModalBody

Компонент ModalBody используется для отображения содержимого модального окна.

ModalFooter

Компонент ModalFooter используется для отображения футера модального окна.

Простой пример

Пример с изменением ширины окна

Вы можете указать ширину модального окна, используя свойство `width`.

Пример с ограничением по высоте

Модальное окно не превышает размеры окна браузера, поэтому содержимое `ModalBody` будет скроллиться.

Пример с отключением обработчиков закрытия

По-умолчанию обработчик `onClose` сработает при нажатии клавиши Esc и при клике за пределы модального окна. Это поведение можно отключить с помощью свойств `disableEscapePressHandler` и `disableBackdropClickHandler`

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import {
  NotificationContainer,
  notification as coreNotification,
  Notification,
} from '@v-uik/notification'
```

```
import { Button } from '@v-uik/button'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Playground } from '@v-uik/notification/examples/Playground'
import RawPlayground from '!!raw-loader!@v-uik/notification/examples/Playgrou
nd'

<Meta
  title={createTitle([COMPONENTS.feedback, 'Notification'])}
  component={NotificationContainer}
/>

<Story
  name="Notification"
>
  {(args) => (
    <>
      <NotificationContainer {...args} />
      <Button onClick={() => coreNotification('notification content')}>
        show notification
      </Button>
    </>
  )}
</Story>
```

NotificationContainer

Компонент NotificationContainer используется для отображения системных нотификаций (сообщений, предупреждений, ошибок и т.п.). Для корректной работы необходимо добавить его в приложение **в одном экземпляре** (например в корень React-дерева наряду с различными провайдерами и т.п.), и затем использовать объект `notification` для отправки сообщений.

```
import { NotificationContainer, notification } from '@v-uik/notification'
```

NotificationOptions

Для отображения сообщений используйте функцию `notification`, первым параметром она принимает содержимое сообщения, а вторым объект настроек (см. ниже). Настройки указанные здесь превалируют над настройками контейнера.

<ArgsTable of={Notification} exclude={[‘isActive’, ‘closeNotification’, ‘removeNotification’]} />

Функция notification имеет встроенные методы, для более удобного вызова различных типов сообщений:

```
notification.success('сообщение')
notification.info('сообщение')
notification.warning('сообщение')
notification.error('сообщение')
```

Также функция notification возвращает идентификатор сообщения, который затем может использоваться для его закрытия программно с помощью метода notification.close()

```
const id = notification('сообщение')
...
notification.close(id)
```

Интерактивный playground

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Popup } from '@v-uik/popup'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Basic } from '@v-uik/popup/examples/Basic'
import RawBasic from '!!raw-loader!@v-uik/popup/examples/Basic'
import { Placement } from '@v-uik/popup/examples/Placement'
import RawPlacement from '!!raw-loader!@v-uik/popup/examples/Placement'

<Meta title={createTitle([COMPONENTS.utility, 'Popup'])} component={Popup} />

<Story
  name="Popup"
  parameters={{
    docs: {
      disable: true,
    },
  }}
  args={{
    open: true,
    children: 'Содержимое popup',
    anchor: () => document.getElementById('anchor'),
    style: {
      padding: 10,
      borderRadius: 4,
      backgroundColor: 'wheat',
    },
  }}
  argTypes={{
    children: {
      description: 'Содержимое компонента',
      control: {
        type: 'text',
      },
    },
    anchor: {
      control: false,
    },
    style: {
      table: {
        disable: true,
```

```
      },
    },
  }}
  decorators={[
    (Story) => (
      <>
        <div
          id="anchor"
          style={{
            margin: '50px 0 50px 150px',
            backgroundColor: 'aqua',
            display: 'inline-flex',
            padding: 10,
            borderRadius: 4,
          }}
        >
          Anchor
        </div>
        <Story />
      </>
    ),
  ]}
>
  {(args) => <Popup {...args} />}
</Story>
```

Popup

Компонент Popup позволяет отображать контент поверх других элементов и позиционировать его относительно какого либо элемента, переданного в свойство anchor. Компонент реализован с помощью сторонней библиотеки Popper.js

import

```
import { Popup } from '@v-uik/popup'
```

```
  <Basic />
```

Позиционирование

Позиционирование Popup регулируется с помощью аттрибута placement.

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { LinearProgress, CircularProgress } from '@v-uik/progress'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { ControlledProgress } from '@v-uik/progress/examples/ControlledProgre
ss'
import RawControlledProgress from '!!raw-loader!@v-uik/progress/examples/Cont
rolledProgress'
import { IndeterminateProgress } from '@v-uik/progress/examples/Indeterminate
Progress'
```

```
import RawCIndeterminateProgress from '!!raw-loader!@v-uik/progress/examples/
IndeterminateProgress'
import { BasicProgress } from '@v-uik/progress/examples/BasicProgress'
import RawBasicProgress from '!!raw-loader!@v-uik/progress/examples/BasicProg
ress'
import { DifferentSizes } from '@v-uik/progress/examples/DifferentSizes'
import RawDifferentSizes from '!!raw-loader!@v-uik/progress/examples/Differen
tSizes'
import { DifferentThickness } from '@v-uik/progress/examples/DifferentThickne
ss'
import RawDifferentThickness from '!!raw-loader!@v-uik/progress/examples/Diff
erentThickness'
import { ControlledCircularProgress } from '@v-uik/progress/examples/Controll
edCircularProgress'
import RawControlledCircularProgress from '!!raw-loader!@v-uik/progress/examp
les/ControlledCircularProgress'
import { CircularWithLabel } from '@v-uik/progress/examples/CircularWithLabel
'
import RawCircularWithLabel from '!!raw-loader!@v-uik/progress/examples/Circu
larWithLabel'
import { DifferentLineColor } from '@v-uik/progress/examples/DifferentLineCol
or'
import RawDifferentLineColor from '!!raw-loader!@v-uik/progress/examples/Diff
erentLineColor'
import { DifferentCircularLineColor } from '@v-uik/progress/examples/Differen
tCircularLineColor'
import RawDifferentCircularLineColor from '!!raw-loader!@v-uik/progress/examp
les/DifferentCircularLineColor'

<Meta
  title={createTitle([COMPONENTS.feedback, 'Progress'])}
  component={LinearProgress}
/>

<Story
  name="Progress"
>
  {(args) => (
    <>
      <div style={{ marginBottom: 100 }}>
        <LinearProgress {...args} />
      </div>
      <CircularProgress {...args} />
    </>
  )}
</Story>
```

Progress

Компоненты отображения прогресса некоторого процесса.

LinearProgress

Линейный прогресс бар.

import
```
import { LinearProgress } from '@v-uik/progress'
```

Неопределенный прогресс-бар

Контролируемый прогресс-бар

Цвета линий

CircularProgress

Круглый прогресс бар.

```
import { CircularProgress } from '@v-uik/progress'
```

Простой

Размеры

Толщина линии

Контролируемый круговой прогресс

С заголовком

Отображение прогресса внутри круга рекомендуется использовать при размере круга `xlg`

Цвета окружности
```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Radio } from '@v-uik/radio'
import { RadioGroup } from '@v-uik/radio-group'
import { LabelControl } from '@v-uik/label-control'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Vertical } from '@v-uik/radio/examples/Vertical'
import RawVertical from '!!raw-loader!@v-uik/radio/examples/Vertical'
import { Horizontal } from '@v-uik/radio/examples/Horizontal'
import RawHorizontal from '!!raw-loader!@v-uik/radio/examples/Horizontal'
import { DifferentPositionRadio } from '@v-uik/radio/examples/DifferentPositi
onRadio'
import RawDifferentPositionRadio from '!!raw-loader!@v-uik/radio/examples/Dif
ferentPositionRadio'
import { DisabledRadio } from '@v-uik/radio/examples/DisabledRadio'
import RawDisabledRadio from '!!raw-loader!@v-uik/radio/examples/DisabledRadi
o'
```

```
<Meta
  title={createTitle([COMPONENTS.controls, 'Radio'])}
  component={RadioGroup}
/>

<Story
  name="Radio"
  argTypes={{
    children: {
      table: {
        disable: true,
      },
    },
  }}
  args={{
    items: [
      {
        value: 'first',
        children: 'first option',
      },
      {
        value: 'second',
        disabled: true,
        children: 'second option',
      },
      {
        value: 'third',
        children: 'third option',
      },
      {
        value: 'fourth',
        children: 'fourth option',
      },
    ],
    value: 'first',
  }}
>
  {({ items, ...args }) => (
    <RadioGroup {...args}>
      {items.map((item, index) => (
        <Radio key={`${index}_${item.value}`} {...item} />
      ))}
    </RadioGroup>
  )}
</Story>
```

Radio

Радиокнопки позволяют пользователю выбрать один вариант из набора.

Radio API

RadioGroup API

imports

```
import { Radio } from '@v-uik/radio'
import { RadioGroup } from '@v-uik/radio-group'
```

Автономная радиокнопка и ее свойства

Радиокнопка — это обертка вокруг `<input type="radio">`, которая обычно используется для создания группы радиокнопок, списков с единичным выбором и тому подобного. Одновременно пользователь может выбрать лишь одну радиокнопку из предложенных.

Далее представлены состояния радиокнопки — не выбрано, выбрано и не активная.

```
  <Radio value="normal" />
  <Radio value="normal" checked />
  <Radio value="active" disabled />
```

Группа радиокнопок

Обертка `RadioGroup` упрощает работу с группой радиокнопок, а также реализует управление радиокнопками с клавиатуры.

Горизонтальная группа

Группа радиокнопок с разным расположением

Заблокированная радиокнопка

Связанные компоненты

- LabelControl

Полезные ссылки

- Доступность

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Select } from '@v-uik/select'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'

import {
  Resize,
  OptionDummy,
  ErrorExample,
  LimitExample,
  CustomOptionProps,
  HelperTextExample,
  MultiSelectExample,
```

```
  ErrorMultiSelectExample,
  MaximumRowsExample,
  MultiSelectHelperText,
} from './examples'

import RawErrorExample from '!!raw-loader!./examples/ErrorExample'
import RawLimitExample from '!!raw-loader!./examples/LimitExample'
import RawCustomOptionProps from '!!raw-loader!./examples/CustomOptionProps'
import RawHelperTextExample from '!!raw-loader!./examples/HelperTextExample'
import RawMultiSelectExample from '!!raw-loader!./examples/MultiSelectExample
'
import RawErrorMultiSelectExample from '!!raw-loader!./examples/ErrorMultiSel
ectExample'
import RawMaximumRowsExample from '!!raw-loader!./examples/MaximumRowsExample
'
import RawMultiSelectHelperText from '!!raw-loader!./examples/MultiSelectHelp
erText'

<Meta
  title={createTitle([COMPONENTS.inputFields, 'Select'])}
  component={Select}
/>

export const options = [
  { value: '', label: 'Выберите опцию' },
  { value: '1', label: 'Опция 1' },
  { value: '2', label: 'Опция 2' },
  { value: '3', label: 'Опция 3' },
  { value: '4', label: 'Опция 4' },
  { value: '5', label: 'Опция 5' },
  { value: '6', label: 'Опция 6' },
  { value: '7', label: 'Опция 7' },
  { value: '8', label: 'Достаточно длинная опция под номером 8' },
  { value: '9', label: 'Достаточно длинная опция под номером 9' },
  { value: '10', label: 'Достаточно длинная опция под номером 10' },
]

<Story
  name="Select"
  args={{
    value: '',
    options: options,
  }}
  decorators={[
    (Story) => (
      <div style={{ height: '150vh' }}>
        <Resize width={300}>
          <Story />
        </Resize>
```

```
      </div>
    ),
  ]}
>
  {(args) => <Select {...args} />}
</Story>
```

Select

Компонент представляет собой раскрывающийся элемент выбора опций. Для отрисовки выпадающего меню используется компонент `Dropdown`, который можно настраивать с помощью свойства `dropdownProps`, и `List`, настраиваемый с помощью `listProps`.

Интерфейс Option

Тип `Option` наследуется от `ListItemProps`, поэтому может принимать свойства компонента `ListItem`.

import

```
import { Select } from '@v-uik/select'
```

Ограничение меню по ширине селекта

Индикация ошибки

Кастомизация опций

Объекты массива `options` принимают свойства компонента `ListItem` для более гибкой настройки.

Вывод подсказки

Для вывода ошибки, достаточно указать `helperText` совместно с флагом `error`

MultiSelect

Мульти-выбор можно включить параметром `multiple`

Индикация ошибки мультиселекта

Ограничение количества отображаемых строк

Количество строк отображаемых за раз можно регулировать числовым параметром `rows`

Вывод подсказки для мультиселекта

Связанные компоненты

- Popup
- InputLabel API

- InputHelperText API

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Slider } from '@v-uik/slider'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { TickDummy } from '@v-uik/slider/examples/Tick.dummy'
import { WithTicks } from '@v-uik/slider/examples/WithTicks'
import RawWithTicks from '!!raw-loader!@v-uik/slider/examples/WithTicks'
import { WithCustomTicks } from '@v-uik/slider/examples/WithCustomTicks'
import RawWithCustomTicks from '!!raw-loader!@v-uik/slider/examples/WithCusto
mTicks'
import { Basic } from '@v-uik/slider/examples/Basic'
import RawBasic from '!!raw-loader!@v-uik/slider/examples/Basic'
import { Disabled } from '@v-uik/slider/examples/Disabled'
import RawDisabled from '!!raw-loader!@v-uik/slider/examples/Disabled'

<Meta title={createTitle([COMPONENTS.controls, 'Slider'])} component={Slider}
/>

<Story
  name="Slider"
  args={{
    value: 15,
    step: 1,
  }}
>
  {(args) => <Slider {...args} />}
</Story>
```

Slider

Компонент выбора значения на числовой прямой

Интерфейс TickItem

import

```
import { Slider } from '@v-uik/slider'
```

Базовый пример

Пример с визуальным отображением возможных значений в виде насечек на прямой

Пример с визуальным отображением пользовательских значений в виде насечек на прямой

Пример с отключением слайдера

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Stepper, Step } from '@v-uik/stepper'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
```

```
import { CustomIcons } from '@v-uik/stepper/examples/CustomIcons'
import RawCustomIcons from '!!raw-loader!@v-uik/stepper/examples/CustomIcons'
import { Mini } from '@v-uik/stepper/examples/Mini'
import RawMini from '!!raw-loader!@v-uik/stepper/examples/Mini'
import { VerticalStepper } from '@v-uik/stepper/examples/VerticalStepper'
import RawVerticalStepper from '!!raw-loader!@v-uik/stepper/examples/Vertical
Stepper'
import { Basic } from '@v-uik/stepper/examples/Basic'
import RawBasic from '!!raw-loader!@v-uik/stepper/examples/Basic'

<Meta
  title={createTitle([COMPONENTS.navigation, 'Stepper'])}
  component={Stepper}
/>

export const items = [
  {
    index: 0,
    children: 'First',
    description: 'First description',
  },
  {
    index: 1,
    children: 'Second',
    description: 'Second description',
  },
  {
    index: 2,
    children: 'Third',
    description: 'Third description',
  },
  {
    index: 3,
    children: 'Fourth',
    description: 'Fourth description',
    error: true,
  },
  {
    index: 4,
    children: 'Fifth',
    description: 'Fifth description',
  },
]

<Story
  name="Stepper"
  argTypes={{
    children: {
      table: {
        disable: true,
```

```
      },
    },
  }}
  args={{
    items: items,
  }}
>
  {({ items, ...args }) => (
    <Stepper {...args}>
      {items.map((item, index) => (
        <Step key={`step_${index}`} {...item} />
      ))}
    </Stepper>
  )}
</Story>
```

Stepper

Компонент используется для отображения пошаговой навигации в некотором процессе.

Step

Для управления отображением каждой конкретного шага используется компонент `Step`

import

```
import { Stepper, Step } from '@v-uik/stepper'
```

Базовый пример

Пример с пользовательскими иконками

Пример с вертикальным расположением

Кастомизация с пользовательскими стилями

```
import {
  Meta,
  Story,
  ArgsTable,
  CFixedColanvas,
  Source,
} from '@storybook/addon-docs'
import { Switch } from '@v-uik/switch'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { SwitchesWithText } from '@v-uik/switch/examples/Examples'
import RawSwitchesWithText from '!!raw-loader!@v-uik/switch/examples/Examples
'
import { DifferentSizeSwitches } from '@v-uik/switch/examples/DifferentSwitch
esSize'
import RawDifferentSizeSwitches from '!!raw-loader!@v-uik/switch/examples/Dif
```

```
ferentSwitchesSize'
import { DifferentPlacement } from '@v-uik/switch/examples/DifferentPlacement
'
import RawDifferentPlacement from '!!raw-loader!@v-uik/switch/examples/Differ
entPlacement'
import { Disabled } from '@v-uik/switch/examples/Disabled'
import RawDisabled from '!!raw-loader!@v-uik/switch/examples/Disabled'
import { LabelControl } from '@v-uik/label-control'

<Meta title={createTitle([COMPONENTS.controls, 'Switch'])} component={Switch}
/>

<Story
  name="Switch"
  args={{
    label: 'Label for Switch',
  }}
  argTypes={{
    label: {
      control: {
        type: 'text',
      },
    },
  }}
>
  {(args) => <Switch {...args} />}
</Story>
```

Switch

Компонент-переключатель поля булева значения

Switch API

LabelControl API

import

```
import { Switch } from '@v-uik/switch'
```

Размер

Переключатели с текстом

Для использования переключателя с текстом, требуется использовать `LabelControl`

Заблокированный переключатель

Расположение

Связанные компоненты

- LabelControl

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Table, TablePagination } from '@v-uik/table'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Basic } from '@v-uik/table/examples/Basic'
import RawBasic from '!!raw-loader!@v-uik/table/examples/Basic'
import { TableLayoutStory } from '@v-uik/table/examples/TableLayoutStory'
import RawTableLayoutStory from '!!raw-loader!@v-uik/table/examples/TableLayo
utStory'
import { ColSpanAndRowSpanStory } from '@v-uik/table/examples/ColSpanAndRowSp
anStory'
import RawColSpanAndRowSpanStory from '!!raw-loader!@v-uik/table/examples/Col
SpanAndRowSpanStory'
import { EventsStory } from '@v-uik/table/examples/EventsStory'
import RawEventsStory from '!!raw-loader!@v-uik/table/examples/EventsStory'
import {
  SmallSizeStory,
  MediumSizeStory,
  LargeSizeStory,
} from '@v-uik/table/examples/SizeStory'
import RawSizeStory from '!!raw-loader!@v-uik/table/examples/SizeStory'
import {
  BorderedNoneStory,
  BorderedRowsStory,
  BorderedBothStory,
} from '@v-uik/table/examples/BorderedStory'
import RawBorderedStory from '!!raw-loader!@v-uik/table/examples/BorderedStor
y'
import { HoverableStory } from '@v-uik/table/examples/HoverableStory'
import RawHoverableStory from '!!raw-loader!@v-uik/table/examples/HoverableSt
ory'
import { MultiRowHeader } from '@v-uik/table/examples/MultiRowHeader'
import RawMultiRowHeader from '!!raw-loader!@v-uik/table/examples/MultiRowHea
der'
import { SortableStory } from '@v-uik/table/examples/SortableStory'
import RawSortableStory from '!!raw-loader!@v-uik/table/examples/SortableStor
y'
import { ExpandableStory } from '@v-uik/table/examples/ExpandableStory'
import RawExpandableStory from '!!raw-loader!@v-uik/table/examples/Expandable
Story.tsx'
import { EmptyDataStory } from '@v-uik/table/examples/EmptyDataStory'
import { ColumnPropsDummy } from '@v-uik/table/examples/ColumnProps.dummy'
import { StripeStory } from '@v-uik/table/examples/StripeStory'
import RawStripeStory from '!!raw-loader!@v-uik/table/examples/StripeStory.ts
x'
import { CustomizeTableComponents } from '@v-uik/table/examples/CustomizeTabl
eComponents'
import RawCustomizeTableComponents from '!!raw-loader!@v-uik/table/examples/C
ustomizeTableComponents.tsx'
import { CustomizedTableCell } from '@v-uik/table/examples/CustomizedTableCel
```

```
l'
import RawCustomizedTableCell from '!!raw-loader!@v-uik/table/examples/Custom
izedTableCell.tsx'
import { PaginateTable } from '@v-uik/table/examples/PaginateTable'
import RawPaginateTable from '!!raw-loader!@v-uik/table/examples/PaginateTabl
e.tsx'
import {
  TreeExpandableStory,
  TreeExpandableCustomizedStory,
} from '@v-uik/table/examples/TreeExpandableStory'
import RawTreeExpandableStory from '!!raw-loader!@v-uik/table/examples/TreeEx
pandableStory.tsx'
import { FixedHeaderStory } from '@v-uik/table/examples/FixedHeaderStory'
import RawFixedHeaderStory from '!!raw-loader!@v-uik/table/examples/FixedHead
erStory.tsx'
import { FixedColumnsStory } from '@v-uik/table/examples/FixedColumnsStory'
import RawFixedColumnsStory from '!!raw-loader!@v-uik/table/examples/FixedCol
umnsStory.tsx'

<Meta
  title={createTitle([COMPONENTS.dataDisplay, 'Table'])}
  component={Table}
/>

<Story
  name="Table"
  args={{
    dataSource: [
      { name: 'Вася', role: 'developer', key: 1 },
      { name: 'Слава', role: 'developer', key: 2 },
      { name: 'Антон', role: 'manager', key: 3 },
      { name: 'Артем', role: 'designer', key: 4 },
      { name: 'Фил', role: 'designer', key: 5 },
    ],
    columns: [
      {
        key: 'name',
        dataIndex: 'name',
        title: 'Имя',
      },
      {
        key: 'role',
        dataIndex: 'role',
        title: 'Роль',
      },
    ],
  }}
>
  {(args) => <Table {...args} />}
</Story>
```

Table

Компонент таблиц.

ColumnProps API

import
```
import { Table } from '@v-uik/table'
```

Обычная таблица с данными

Разделители ячеек

Ячейки таблицы могут быть визуально разделены тремя способами:

Разделение строк

Вариант по-умолчанию. Можно указать явно с помощью свойства `bordered` со значением `rows`.

Разделение строк и столбцов

Выставив свойству `bordered` значение `rows-columns` можно добиться более строгого вида таблицы.

Без разделителей

Можно убрать границы ячеек, указав значение `none` для свойства `bordered`. В этом случае рекомендуется “облегчить” заголовок таблицы изменением цвета заливки.

Размеры таблиц

Таблицы имеют три размера ячеек, которые используются в зависимости от необходимой плотности данных.

Стандартный размер (medium)

Подходит для большинства случаев. Вы можете не передавать свойство `size` и тогда значение `medium` будет выставлено по умолчанию или явно передать `size="medium"`.

Таблицы повышенной плотности (small)

Если нужно вывести много данных, можно использовать таблицы высокой плотности. Для этого установите `size="small"`.

Разреженная таблица (large)

Широкие ячейки, чтобы снизить когнитивную нагрузку при работе с табличными данными.

Полосатая таблица

Свойство `stripe`, включает оформление полосатой таблицы, где выделяется каждая четная строка.

Индикация строк при наведении курсора

Свойство `hoverable` включает индикацию строки при наведении на нее курсора

Сложный (многострочный) заголовок

Для создания многострочных заголовков таблицы в объектах массива `columns` можно указать свойство `children`, содержащее массив объектов этого же типа.

colSpan и rowSpan

Если вам требуется объединить ячейки по вертикали или горизонтали, вы можете воспользоваться атрибутами `colSpan` и `rowSpan`, которые можно задать для каждой ячейки с помощью функции `setCellProps`. Для этого необходимо вернуть объект с полями `colSpan` и `rowSpan`. Чтобы не отображать ячейку, верните для каждого свойства `0`.

Таблица с фиксированным размером ячеек

С помощью свойства `tableLayout` вы можете зафиксировать размер ячеек в таблице. Также объекты массива `columns` принимают свойство `width` в виде числа пикселей для указания конкретной ширины столбца.

Обработка событий таблицы, строк и ячеек

*события выводятся через console.log() в консоль браузера

Сортировка

Сортировку можно разделить на 2 части — отображение текущего состояния сортировки и манипуляции с данными. Таблица реализует первую часть — хранит состояние и генерирует события с полезной нагрузкой необходимые для реализации второй части.

Почему вы не сделали манипуляции с данными внутри таблицы?

Пользователи по разному используют таблицу — кто-то сортирует данные с учетом non-ASCII characters, кому-то требуется мульти-сортировка по нескольким столбцам, а кто-то и вовсе не делает сортировку на фронте, а делает запрос на бек с набором параметров. Чтобы покрыть большинство кейсов, мы стараемся предоставить необходимый набор инструментов, без перегрузки API большим набором опций.

Расширяемая строка

Если объект конфигуратора столбцов `columns` содержит свойство `kind` со значением `expand`, то для этой строки включается функциональность “расширения”.

Свойство `isRowExpanded` управляет отображением состояния расширяемой строки — если переданная функция обратного вызова вернет `true`, строка раскрыта, если `false` — закрыта.

Вы можете скрыть отображение ячейки для расширяемой строки (только иконку, сама ячейка останется в DOM дереве) — для этого в качестве значения для свойства `isRowExpanded` передайте `undefined` или `null`.

Таблица с древовидной структурой (TreeData)

Для отображения таблицы с древовидной структурой (в которой строки могут иметь подстроки и показывать/скрывать их по необходимости), необходимо чтобы объекты `dataSource` содержали поле `children`, в котором должен находиться массив таких же объектов. Также требуется, чтобы объект конфигуратора столбцов `columns` содержал поле `kind` со значением `tree` и поле `isRowExpanded`, которое является функцией, вызываемой для каждой ячейки этого столбца. Эта функция должна возвращать булеву переменную, указывающую раскрыта строка или нет.

На данный момент реализован только контролируемый вариант такой таблицы, то есть вам придется самостоятельно обрабатывать состояние открытых/закрытых строк. В этом вам поможет обработка события с типом `treeExpand` в обработчике onChange (см. пример ниже)

Кастомизированный вариант:

Таблица с фиксированной шапкой

Если расположить таблицу в скроллящемся контейнере, можно зафиксировать ее header при скролле, используя свойство `fixedHeader`. Эта функциональность реализована с помощью `position: sticky`, поэтому в IE 11 она работать не будет.

Таблица с фиксированными столбцами

Чтобы зафиксировать столбец при горизонтальном скролле, необходимо добавить в соответствующий объект columns поля `fixed` (со значением `start` или `end`, определяющим сторону фиксации) и `width` (ширина колонки в пикселях). Для корректного отображения остальных колонок, рекомендуется задать ширину либо каждой колонке по-отдельности, либо таблице целиком. Функциональность основана на css-свойстве `position: sticky`, поэтому в IE 11 она работать не будет. Также, на данный момент, эта функциональность не работает со сложным (многострочным) заголовком таблицы.

Таблица с переопределенными компонентами

С помощью свойства `components` вы можете переопределить как будут отрисованы элементы таблицы.

Таблица с пользовательским содержимом ячеек

В объекте описания колонки `columns`, с помощью функции `renderCellContent`, вы можете указать как ячейке рендерить ее содержимое.

Таблица с пустыми данными

Таблица с пагинацией

Для пагинации таблицы существует компонент `TablePagination` импортируемый из `@v-uik/table`. Расположение сверху/снизу регулируется последовательностью компонентов.

Pagination API

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Tabs, Tab } from '@v-uik/tabs'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Disabled } from '@v-uik/tabs/examples/Disabled'
import RawDisabled from '!!raw-loader!@v-uik/tabs/examples/Disabled'
import { Filled } from '@v-uik/tabs/examples/Filled'
import RawFilled from '!!raw-loader!@v-uik/tabs/examples/Filled'
import { WithIcons } from '@v-uik/tabs/examples/WithIcons'
import RawWithIcons from '!!raw-loader!@v-uik/tabs/examples/WithIcons'
import { WithRemoving } from '@v-uik/tabs/examples/WithRemoving'
import RawWithRemoving from '!!raw-loader!@v-uik/tabs/examples/WithRemoving'
import { Vertical } from '@v-uik/tabs/examples/Vertical'
import RawVertical from '!!raw-loader!@v-uik/tabs/examples/Vertical'

<Meta title={createTitle([COMPONENTS.navigation, 'Tabs'])} component={Tabs} /
>

export const data = [
  {
    value: '1',
    header: '1 вкладка',
    children: 'Содержимое 1 вкладки',
  },
  {
    value: '2',
    header: '2 вкладка',
    children: 'Содержимое 2 вкладки',
  },
  {
    value: '3',
    header: '3 вкладка',
    children: 'Содержимое 3 вкладки',
  },
  {
    value: '4',
```

```
    header: '4 вкладка',
    children: 'Содержимое 4 вкладки',
  },
]

<Story
  name="Tabs"
  args={{
    items: data,
    value: '1',
  }}
>
  {({ items, ...args }) => (
    <Tabs {...args}>
      {items.map((item, index) => (
        <Tab key={`${index}_${item.value}`} {...item} />
      ))}
    </Tabs>
  )}
</Story>
```

Tabs

Компонент для отображения элементов, объединенных в секции

Tab

Для управления отображением каждой конкретной секции используется компонент Tab

import

```
import { Tabs, Tab } from '@v-uik/tabs'
```

Вариант с заливкой

Отключаемые секции

Секцию можно запретить для выбора, добавив ей атрибут disabled

Вариант с иконками и бейджем

Вариант с добавлением/удалением табов

Вертикальное расположение

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Tag } from '@v-uik/tag'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Basic } from '@v-uik/tag/examples/Basic'
import RawBasic from '!!raw-loader!@v-uik/tag/examples/Basic'
import { ColorTagsExample } from '@v-uik/tag/examples/ColorTagsExample'
```

```
import RawColorTagsExample from '!!raw-loader!@v-uik/tag/examples/ColorTagsExample'
import { DifferentSizeExample } from '@v-uik/tag/examples/DifferentSizeExample'
import RawDifferentSizeExample from '!!raw-loader!@v-uik/tag/examples/DifferentSizeExample'
import { TagTypesExample } from '@v-uik/tag/examples/TagTypesExample'
import RawTagTypesExample from '!!raw-loader!@v-uik/tag/examples/TagTypesExample'
import { CustomColor } from '@v-uik/tag/examples/CustomColor'
import RawCustomColor from '!!raw-loader!@v-uik/tag/examples/CustomColor'
import { ColorIndicator } from '@v-uik/tag/examples/ColorIndicator'
import RawColorIndicator from '!!raw-loader!@v-uik/tag/examples/ColorIndicator'

<Meta title={createTitle([COMPONENTS.controls, 'Tag'])} component={Tag} />

<Story
  name="Tag"
  args={{
    children: 'Tag',
  }}
  argTypes={{
    children: {
      description: 'Содержимое компонента',
      control: {
        type: 'text',
      },
    },
  }}
>
  {(args) => <Tag {...args} />}
</Story>
```

Tag

Tag - это компактные элементы, представляющие входные данные, атрибут или действие. В зависимости от назначения, они делятся на 3 типа, каждый из которых обладает собственными атрибутами.

import

```
import { Tag } from '@v-uik/tag'
```

Типы тегов

Компонент может быть одним из четырех типов: `lite`, `secondary`, `primary` и `color`. Теги с типом `color` выполняют лишь визуальную функцию и не могут быть интерактивными (игнорируют значения свойств `onClick`, `onDelete` и `selected`).

Размеры тегов

Цветовые теги

Интерактивные теги

Все теги, за исключением типа `color`, могут управляться с помощью свойств `selected`, `onClick`, `onDelete`

Кастомизация интерактивных тегов

Немного стилизовав теги с помощью CSS, можно добиться пользовательской цветовой схемы.

Дополнительная цветовая индикация тегов

Альтернативным вариантом кастомизации палитры тегов может послужить добавление цветного индикатора.

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { TagInput } from '@v-uik/tag'

import { createTitle, COMPONENTS } from '../../docs/showroom/config'

import { BasicTagInput } from './examples/BasicTagInput'
import RawBasic from '!!raw-loader!./examples/BasicTagInput'

<Meta
  title={createTitle([COMPONENTS.controls, 'TagInput'])}
  component={TagInput}
/>

<Story
  name="TagInput"
  args={{
    children: 'Add tag',
    placeholder: 'new tag',
  }}
>
  {(args) => <TagInput {...args} />}
</Story>
```

TagInput

TagInput - элемент для добавления новых тегов

import

```
import { TagInput } from '@v-uik/tag'
```

Стандартное поле ввода

Размеры тэга
```
  <div style={{ display: 'flex', alignItems: 'center', gap: 16 }}>
    <TagInput size="sm">small</TagInput>
    <TagInput size="md">medium</TagInput>
  </div>
```

Поле заблокировано для ввода
```
  <TagInput disabled>Добавить тег</TagInput>

import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Textarea } from '@v-uik/textarea'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Basic } from '@v-uik/textarea/examples/Basic'
import RawBasic from '!!raw-loader!@v-uik/textarea/examples/Basic'
import { Disabled } from '@v-uik/textarea/examples/Disabled'
import RawDisabled from '!!raw-loader!@v-uik/textarea/examples/Disabled'
import { WithError } from '@v-uik/textarea/examples/WithError'
import RawWithError from '!!raw-loader!@v-uik/textarea/examples/WithError'
import { FullWidth } from '@v-uik/textarea/examples/FullWidth'
import RawFullWidth from '!!raw-loader!@v-uik/textarea/examples/FullWidth'
import { WithRows } from '@v-uik/textarea/examples/WithRows'
import RawWithRows from '!!raw-loader!@v-uik/textarea/examples/WithRows'

<Meta
  title={createTitle([COMPONENTS.inputFields, 'Textarea'])}
  component={Textarea}
/>

<Story
  name="Textarea"
>
  {(args) => <Textarea {...args} />}
</Story>
```

Textarea

Компонент ввода многострочного текста

import
```
import { Textarea } from '@v-uik/textarea'
```

Стандартное поле ввода

export const onChange = () => {}

Поле заблокировано для ввода

Поле содержит ошибку

Растягивание поля на всю ширину контейнера

Изменение высоты поля

В этом примере задается количество строк с помощью свойства rows.

```
import { Meta, Story, Canvas, ArgsTable, Source } from '@storybook/addon-docs
'
import { Tooltip } from '@v-uik/tooltip'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { Multi } from '@v-uik/tooltip/examples/Multi'
import RawMulti from '!!raw-loader!@v-uik/tooltip/examples/Multi'
import { Single } from '@v-uik/tooltip/examples/Single'
import RawSingle from '!!raw-loader!@v-uik/tooltip/examples/Single'
import { Placement } from '@v-uik/tooltip/examples/Placement'
import RawPlacement from '!!raw-loader!@v-uik/tooltip/examples/Placement'
import { Interactive } from '@v-uik/tooltip/examples/Interactive'
import RawInteractive from '!!raw-loader!@v-uik/tooltip/examples/Interactive'

<Meta
  title={createTitle([COMPONENTS.feedback, 'Tooltip'])}
  component={Tooltip}
/>

export const decorators = [
  (Story) => (
    <div style={{ marginTop: 100, marginLeft: 200, display: 'inline-flex' }}>
      <Story />
    </div>
  ),
]

<Story
  name="Tooltip"
  decorators={decorators}
  args={{
    children: (
      <div style={{ backgroundColor: 'lightgrey', width: 60, height: 60 }}>
        hover me
      </div>
    ),
    dropdownProps: {
      placement: 'right',
      content: 'Tooltip content',
    },
  }}
```

```
  argTypes={{
    children: {
      description: 'Содержимое компонента',
      control: {
        disable: true,
      },
    },
  }}
>
  {(args) => <Tooltip {...args} />}
</Story>
```

Tooltip

Tooltip - всплывающие подсказки, появляющиеся когда пользователь наводит курсор на элемент, фокусируется на нем или нажимает на него.

import

```
import { Tooltip } from '@v-uik/tooltip'
```

Подсказки, поддерживающие несколько строк (default)

Компактное отображение содержимого подсказок в одну строку

Компактное отображение подсказок выводит содержимое с уменьшенными отступами, а текст выводится одной строкой.

Позиционирование подсказок

Позиционирование подсказки регулируется с помощью аттрибута placement.

Вариант с интерактивными элементами

Во избежание слияния цветов компонентов в гамме `secondary` с фоном тултипа, можно использовать темную тему или стилизовать, используя палитру токенов `theme.sys.color.inverseOnBackground*`. Также внутри тултипа можно обращаться к `TooltipContext`, который содержит метод `close()` для его закрытия, если вы работаете с неконтролируемым состоянием (без контроля свойства open).

Связанные компоненты

- Dropdown

```
import { Meta, Story, ArgsTable, Canvas, Source } from '@storybook/addon-docs
'
import { Text } from '@v-uik/typography'
import { createTitle, COMPONENTS } from '../../docs/showroom/config'
import { BasicExample } from '@v-uik/typography/examples/BasicExample'
import RawBasicExample from '!!raw-loader!@v-uik/typography/examples/BasicExa
mple'

<Meta
```

```
  title={createTitle([COMPONENTS.dataDisplay, 'Typography'])}
  component={Text}
/>

<Story
  name="Typography"
  args={{
    children: 'Text',
  }}
  argTypes={{
    children: {
      description: 'Содержимое компонента',
      control: {
        type: 'text',
      },
    },
  }}
>
  {(args) => <Text {...args} />}
</Story>
```

Text

Компонент для отображения текста

import

```
import { Text } from '@v-uik/typography'
```

Примеры использования

Компонент «Типографика» позволяет легко применить в приложении набор значений толщины и размера шрифта по умолчанию.

Часто встречающиеся проблемы и пути их устранения

В компоненте отсутствует накопленная база негативных прецедентов.