

Пользовательский интерфейс

Организация интерфейса

- Управление без сложностей
Простые жесты,
Взаимодействие со всем экраном
- Удобная многозадачность
- Атмосферы управляют внешним видом
- Приложения используют весь экран



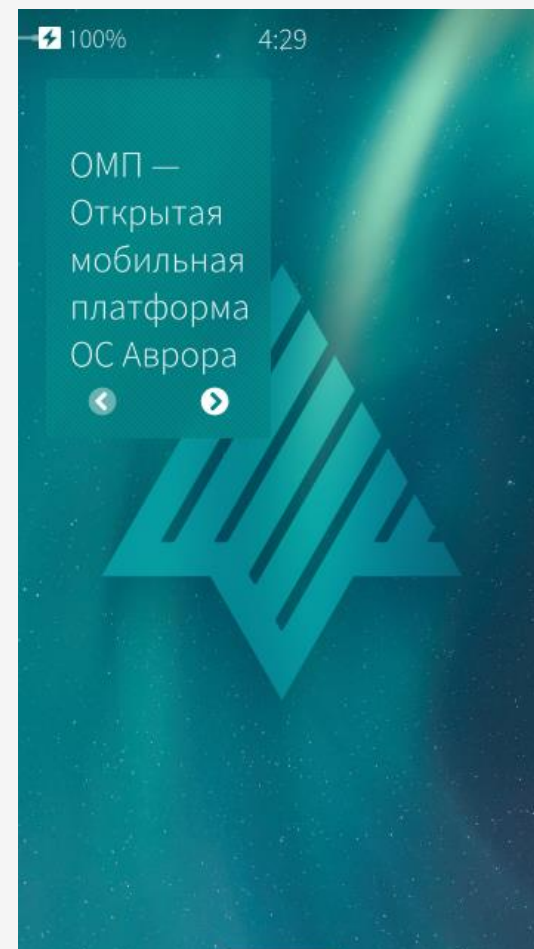
Экран блокировки

- **Статус устройства**
 - Время и дата
 - Важная информация о состоянии устройства — заряд батареи, уровень связи и т. д.
 - Уведомления
- **Быстрый доступ**
 - Камера
 - Выбранные приложения
- **Разблокировка экрана свайпом слева направо**



Домашний экран

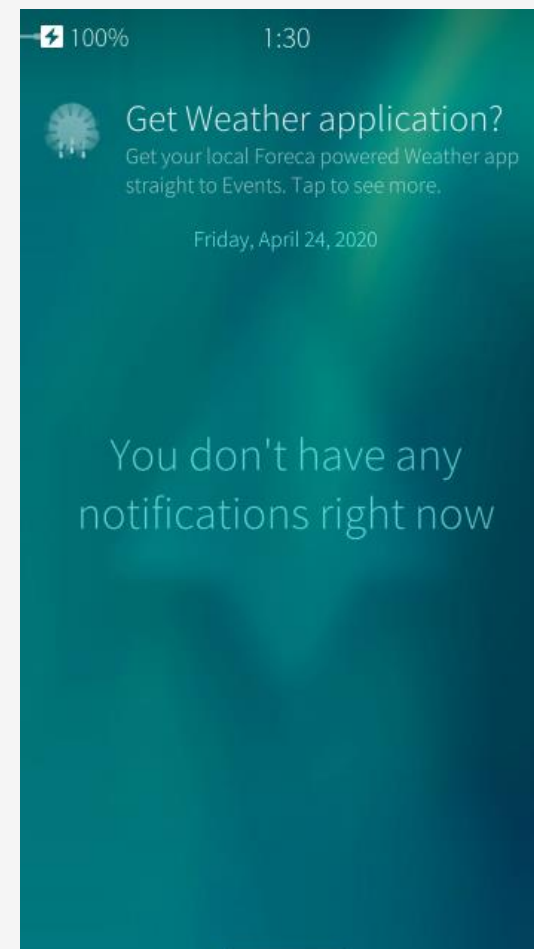
- **Появляется**
 - После разблокировки устройства
 - После сворачивания приложения
- **Содержит обложки (Cover)** — миниатюры приложений
 - Важна информацию о приложении —
 - статус или необходимые данные
 - Позволяет взаимодействовать с приложением
 - При нажатии приложение разворачивается



События

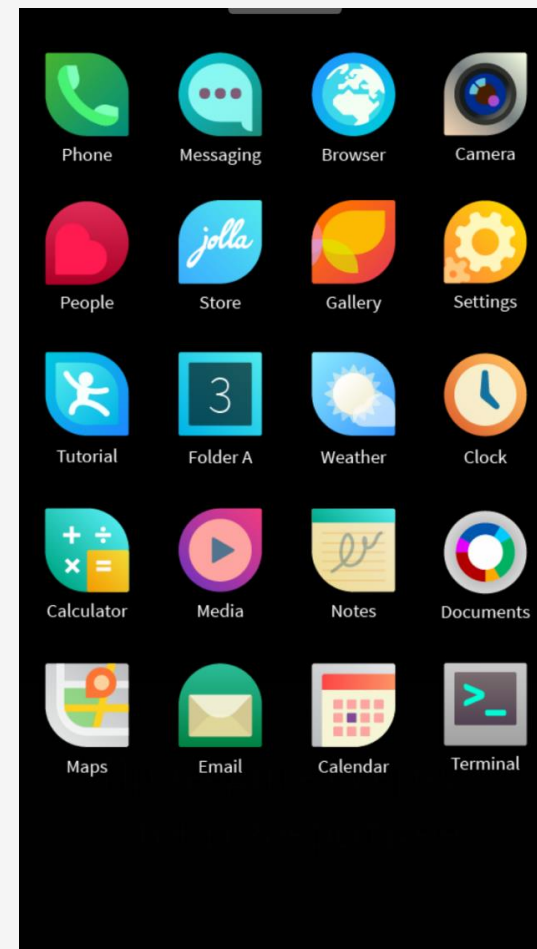
Отображает информацию о событиях

- Уведомления приложений
- Погода
- Календарь



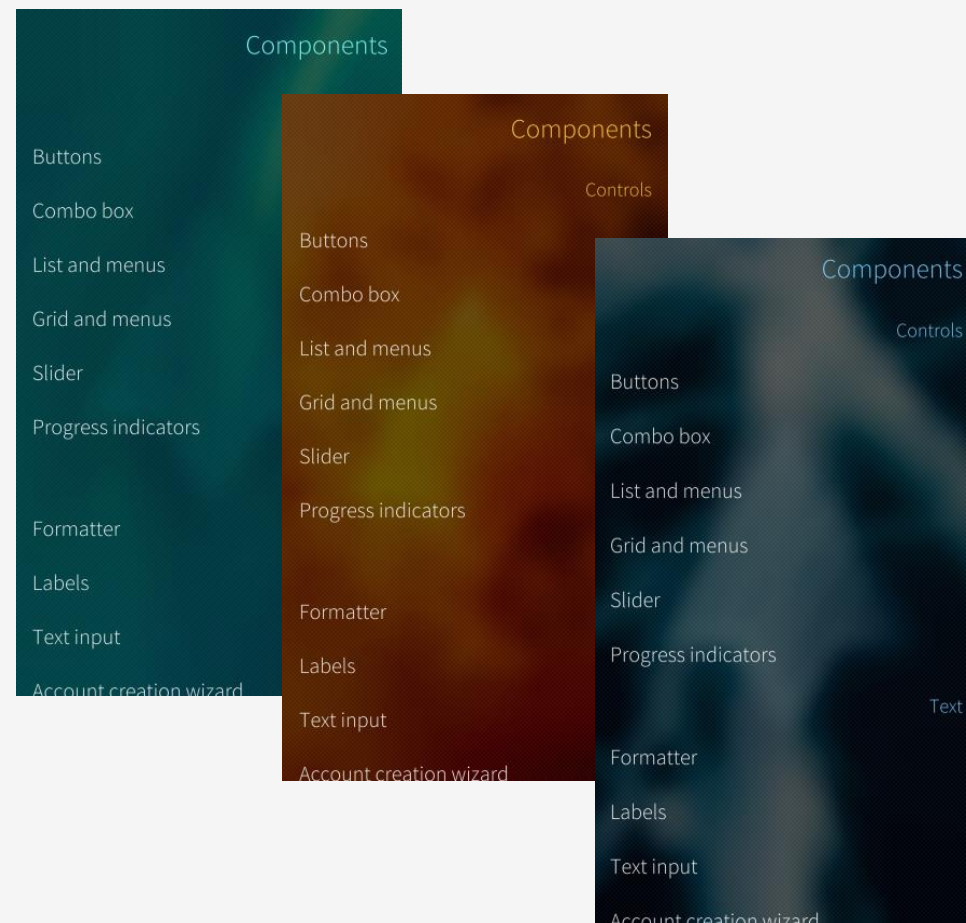
Панель приложений

- Содержит все установленные приложения
- Иконки могут быть перемещены, сгруппированы в папки
- Приложение удаляется долгим нажатием на иконку приложения
- Открывается с любого экрана свайпом снизу вверх

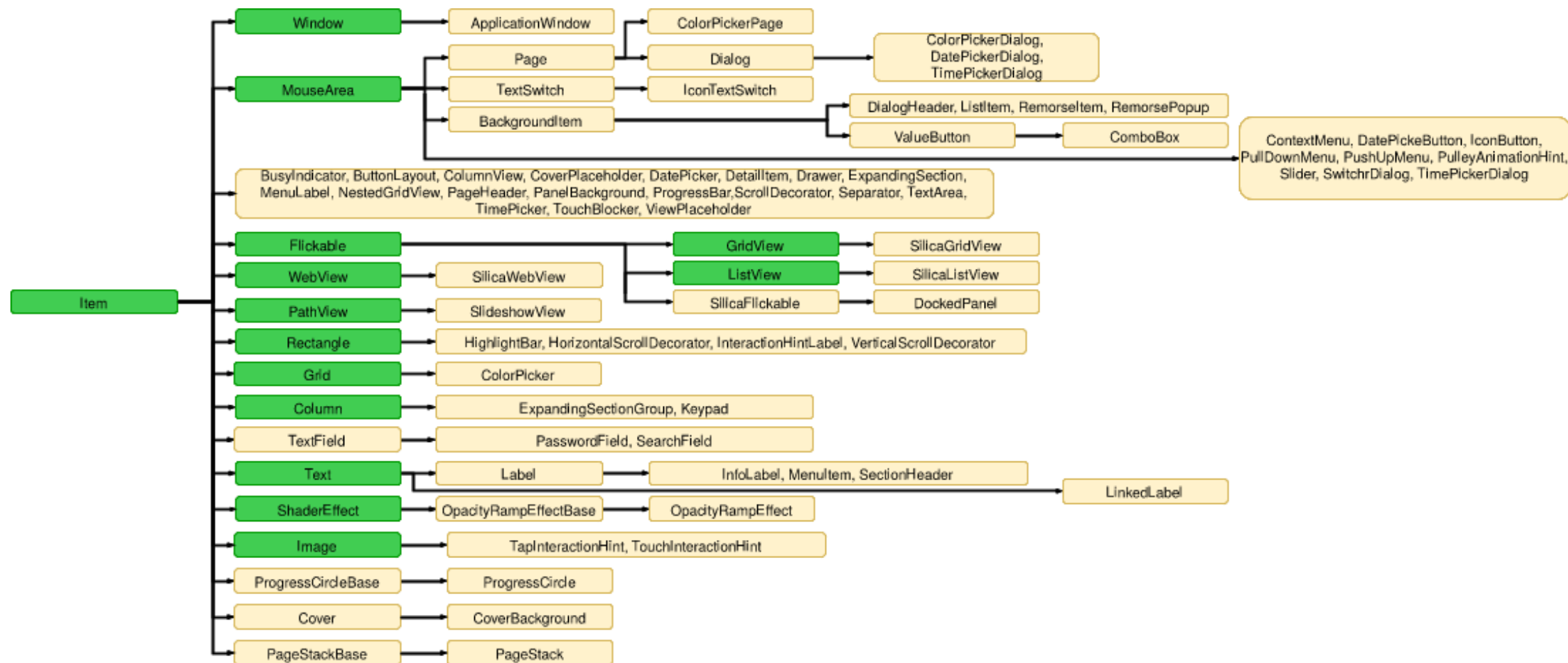


Атмосфера

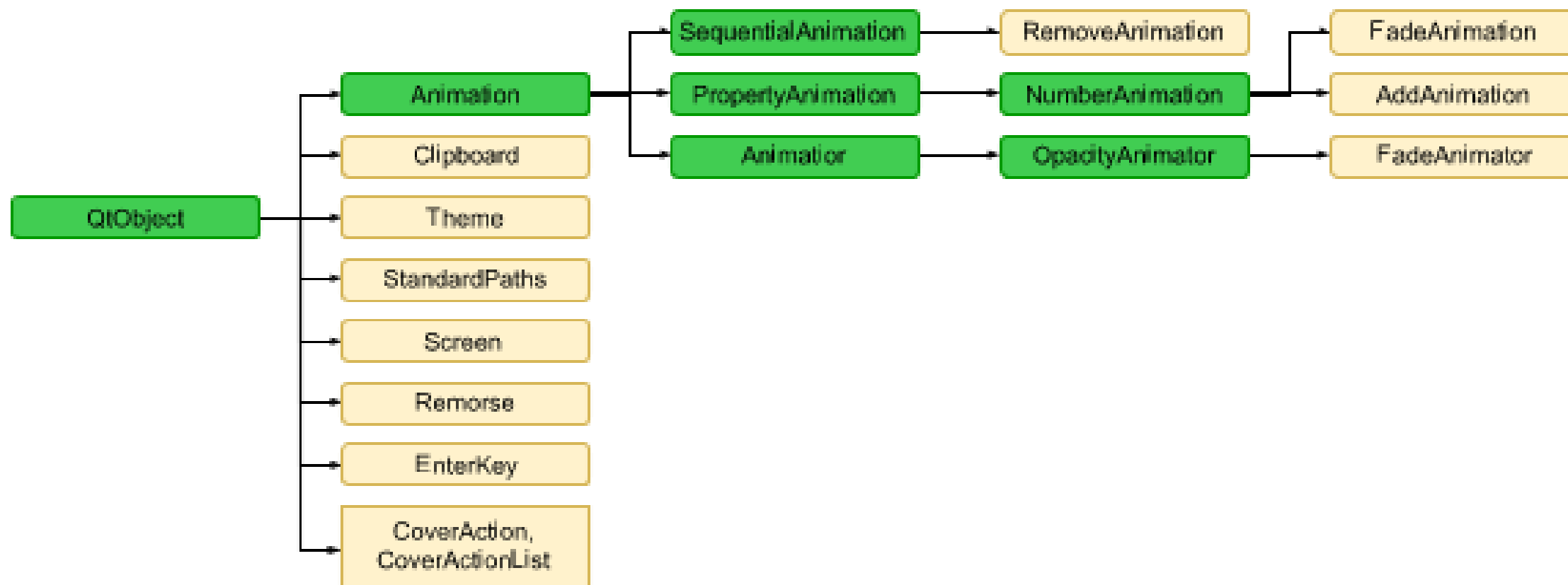
- Определяет стиль UI
- Обои
- Цвет темы
- Звуки темы
- Атмосферу можно создать вручную из изображения
- Приложения по умолчанию соответствуют атмосфере



Визуальные элементы Silica



Невизуальные элементы Silica



ApplicationWindow – точка входа

- **initialPage** : **var** — какая страница будет загружена при запуске
- **background.*** — настройки фона (цвет, изображение и т.п.)
- **cover** : **var** — определяет обложку приложения
- **pageStack** : **PageStack** — стек отображаемых страниц,
- **allowedOrientations** : **enumeration** — набор доступных ориентаций экрана

Orientation.All

Orientation.PortraitMask, **Orientation.Portrait**, **Orientation.PortraitInverted**

(не для смартфонов)

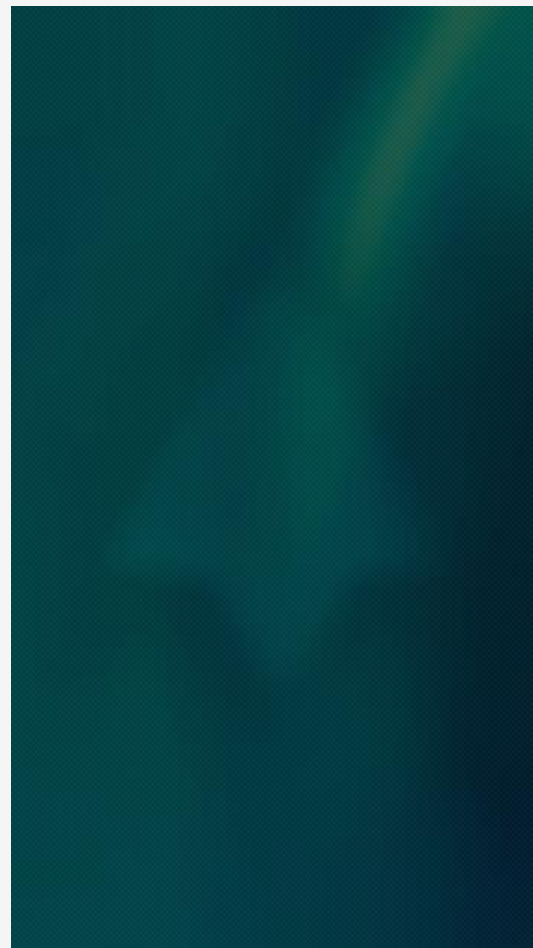
Orientation.LandscapeMask, **Orientation.Landscape**, **Orientation.LandscapeInverted**

- **activate()** — вывести приложение в полноэкранный режим
- **deactivate()** — свернуть приложение

ApplicationWindow – точка входа

```
import QtQuick 2.0
import Sailfish.Silica 1.0

ApplicationWindow {
    initialPage:
        Component { Page {} }
    cover:
        Component { CoverBackground {
    }}
    allowedOrientations:
        defaultAllowedOrientations
}
```



Page – страница приложения

Ориентация

`allowedOrientations` : `enumeration` — набор доступных ориентаций экрана
`isLandscape`, `isPortrait` : `bool` — статус текущей ориентации

Навигация

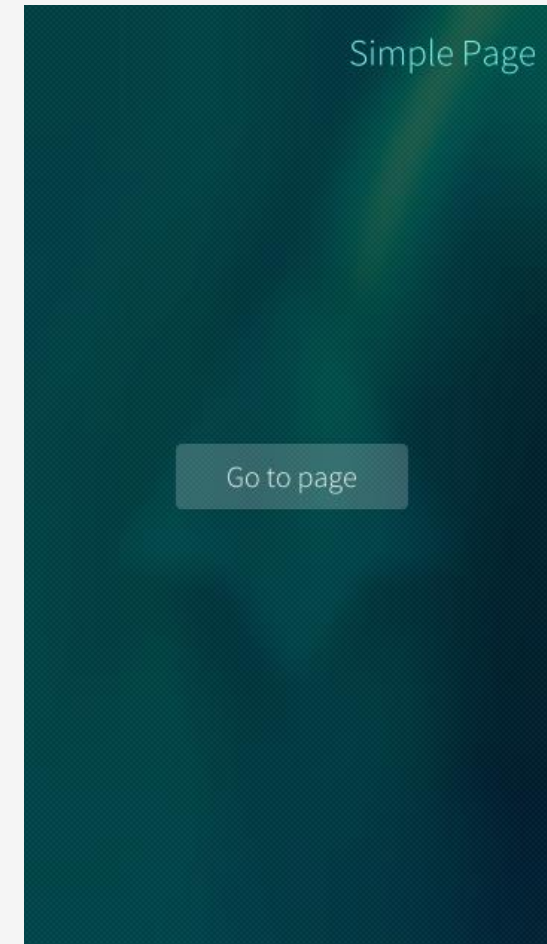
`canNavigateForward` : `bool` — можно ли перейти к следующей странице
`forwardNavigation` : `bool` — управление переходом к следующей странице
`backNavigation` : `bool` — управление переходом к предыдущей странице
`navigationStyle` : `enumeration` — вертикальная или горизонтальная навигация

`status` : `enumeration`

`PageStatus.Inactive` — страница не активна в стеке страниц и не отображается
`PageStatus.Activating` — страница переходит в активное состояние
`PageStatus.Active` — текущая активная страница
`PageStatus.Deactivating` — страница переходит в неактивное состояние

Page – страница приложения

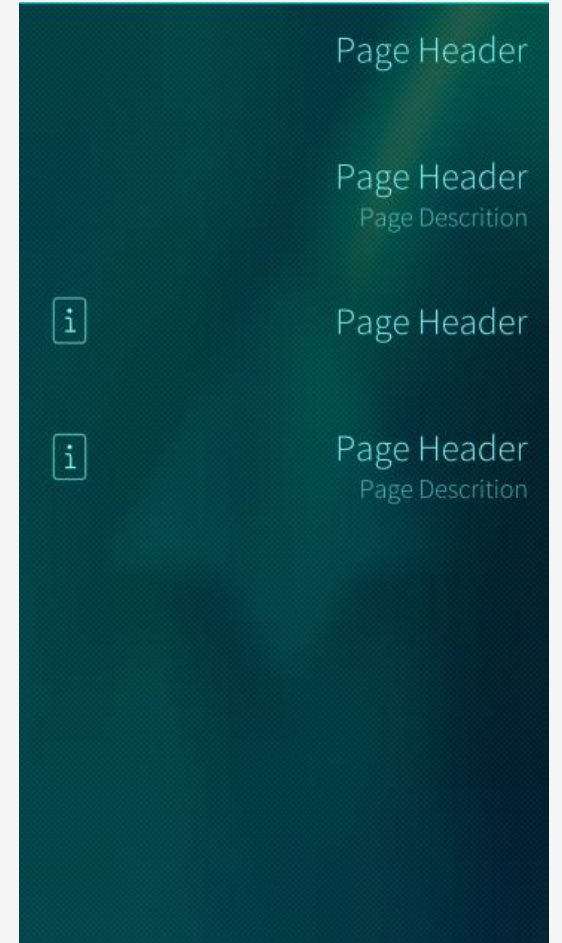
```
Page {
    allowedOrientations: Orientation.All
    PageHeader { title: qsTr("Simple Page") }
    Button {
        anchors.centerIn: parent
        text: "Go to page"
        onClicked: pageStack.push("SecondPage.qml")
    }
    onStatusChanged: {
        switch (status) {
        case PageStatus.Inactive:
            return console.log("Inactive");
        case PageStatus.Activating:
            return console.log("Activating");
        case PageStatus.Active:
            return console.log("Active");
        case PageStatus.Deactivating:
            return console.log("Deactivating");
        }
    }
}
```



PageHeader – заголовок страницы

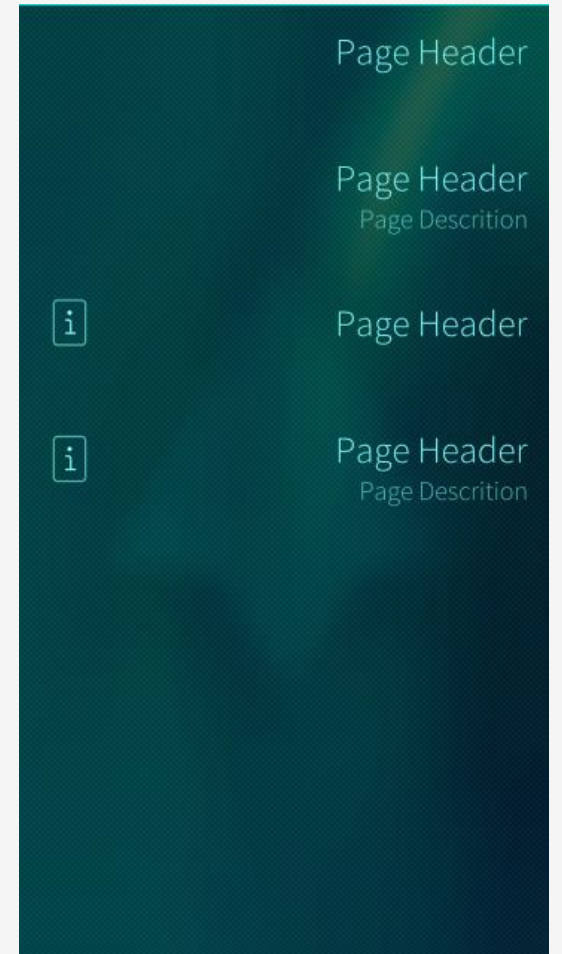
```
PageHeader {  
  title: qsTr("Page Header")  
}
```

```
PageHeader {  
  title: qsTr("Page Header")  
  description: qsTr("Page Description")  
}
```



PageHeader – заголовок страницы

```
PageHeader {  
  title: qsTr("Page Header")  
  extraContent.children: [  
    Image {  
      anchors.verticalCenter: parent.verticalCenter  
      source: "image://theme/icon-m-about?" +  
        Theme.highlightColor  
    }  
  ]  
}  
  
PageHeader {  
  title: qsTr("Page Header")  
  description: qsTr("Page Description")  
  extraContent.children: [ Image {  
    anchors.verticalCenter: parent.verticalCenter  
    source: "image://theme/icon-m-about?" +  
      Theme.highlightColor  
  }  
]
```



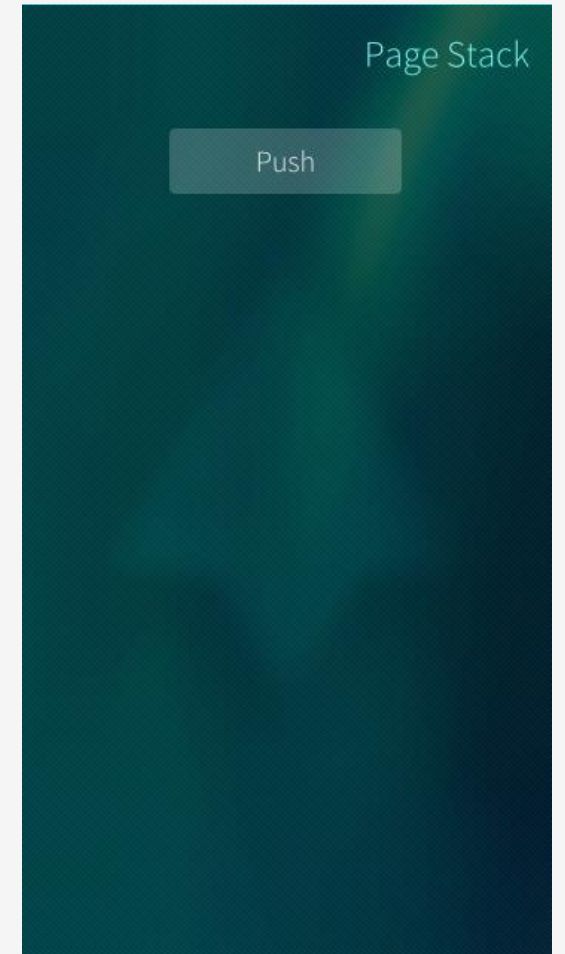
PageStack – контроль навигации

- **busy** : **bool** — true, если происходит анимация перехода от экрана к экрану
- **currentPage** : **Item** — страница на вершине стека
- **depth** : **int** — текущее количество страниц на стеке
- **push()** — добавить указанную страницу на вершину стека
- **pop()** — извлечь одну или несколько страниц из стека
- **pushAttached()** — прикрепить указанную страницу к странице, находящейся на вершине стека
- **popAttached()** — извлечь одну или несколько прикрепленных страниц из стека
- **navigateBack()** — переместиться на предыдущую страницу в стеке
- **navigateForward()** — переместиться на следующую страницу в стеке
- **clear()** — очистить стек страниц

Операции PageStack: Push и Pop

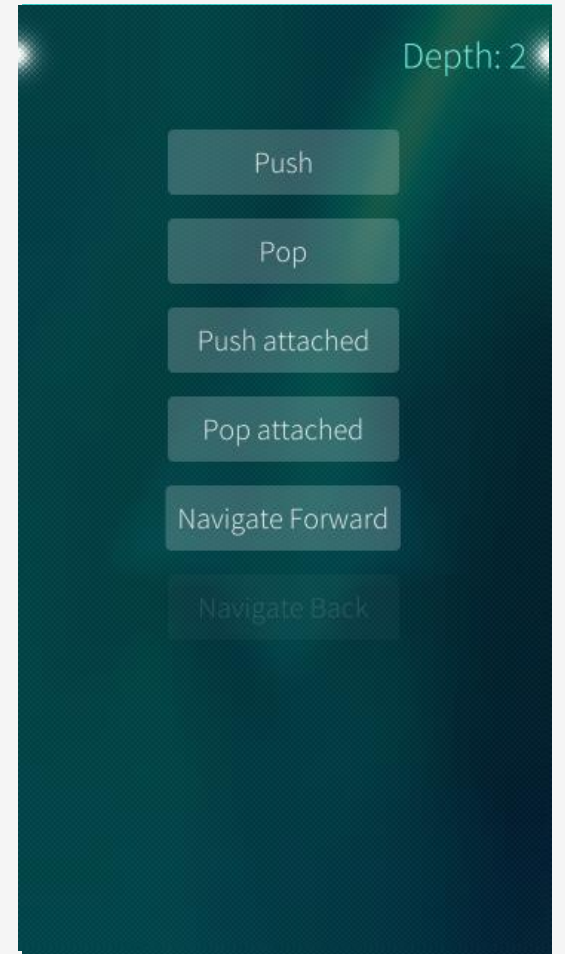
```
Button {  
    text: "Push"  
    onClicked: pageStack.push(  
        Qt.resolvedUrl("SecondPage.qml")  
    )  
}
```

```
Button {  
    text: "Pop"  
    onClicked: pageStack.pop()  
}
```



Операции PageStack: PushAttached

```
Button {  
    text: "Push attached"  
    onClicked: pageStack.pushAttached(  
        Qt.resolvedUrl("SecondPage.qml"))  
}  
  
Button {  
    text: "Navigate Forward"  
    onClicked: pageStack.navigateForward()  
}  
  
Button {  
    text: "Navigate Back"  
    onClicked: pageStack.navigateBack()  
}
```



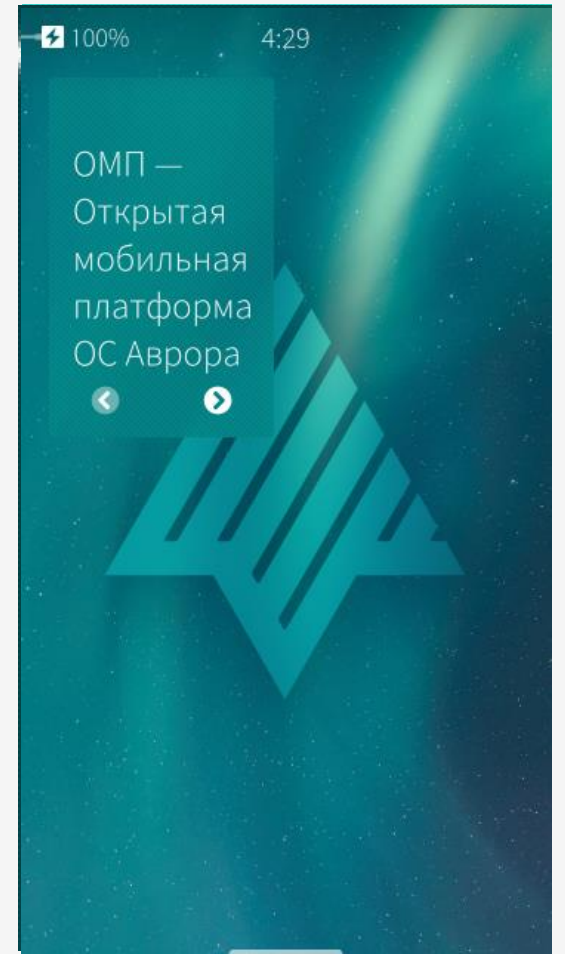
Cover – обложка приложения

Отображает информацию и позволяет управлять приложением без необходимости его разворачивать

- **Cover** – обложка приложения на домашнем экране
 - `CoverBackground` — полупрозрачная обложка
 - `coverActionList` — действия обложки
- **ApplicationWindow**
 - `cover : var` — определяет обложку

Cover – обложка приложения

```
CoverBackground {  
  Label {  
    width:parent.width  
    text: currentPageTitle  
    truncationMode: TruncationMode.Elide  
    font.pixelSize: Theme.fontSizeLarge  
    wrapMode: Label.WordWrap  
  }  
  CoverActionList {  
    CoverAction {  
      iconSource: "image://theme/icon-cover-previous"  
      onTriggered: navigateBack()  
    }  
    CoverAction {  
      iconSource: "image://theme/icon-cover-next"  
      onTriggered: navigateForward()  
    }  
  }  
}
```



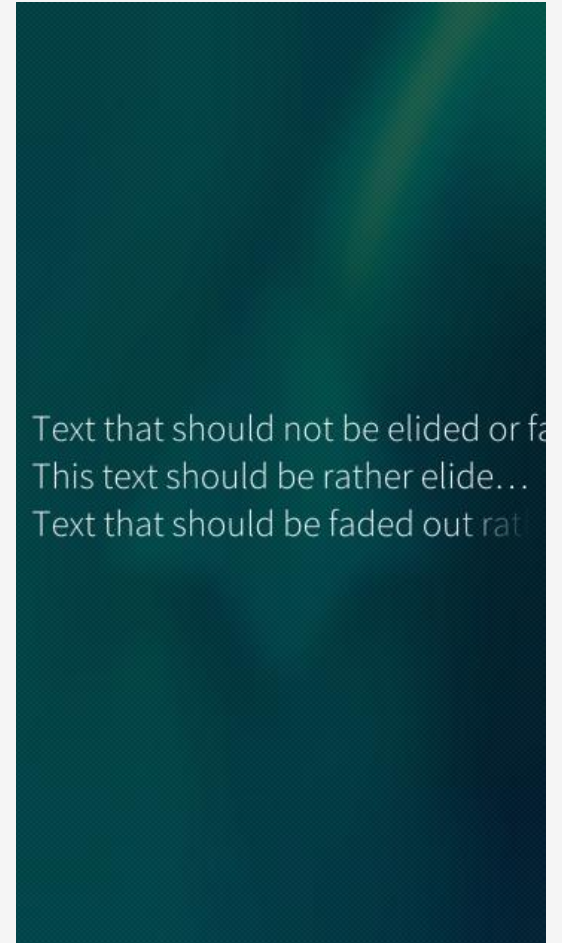


Label – вывод текста

- Унаследован от `Text`
- `highlighted` : `bool` — подсветка элемента
- `palette` : `Palette` — цветовая палитра
- `truncationMode` : `enumeration` — способ обрезать строку, если текст не помещается в размеры элемента

Label – вывод текста

```
Label {  
  text: qsTr("Text that should not be elided or faded")  
  font.pixelSize: Theme.fontSizeMedium  
}  
Label {  
  text:  
    qsTr("This text should be elided off the right end")  
  truncationMode: TruncationMode.Elide  
  font.pixelSize: Theme.fontSizeMedium  
}  
Label {  
  text:  
    qsTr("Text that should be faded out rather than elided")  
  truncationMode: TruncationMode.Fade  
  font.pixelSize: Theme.fontSizeMedium  
}
```



LinkedLabel – форматируемый текст

- **Текстовый абзац со встроенным синтаксическим анализатором:**
 - URL-адреса
 - Номера телефонов
 - Адреса электронной почты
- **defaultLinkActions** : **bool** — управление обработкой нажатий на ссылки
- **onLinkActivated()** — сигнал нажатия на ссылку
- **plainText** : **string** — текст без тэгов
- **shortenUrl** : **bool** — сокращать ли URL-адреса в тексте "http://foobar.com"
→ "foobar.com"

LinkedList – форматированный текст

```
LinkedList {  
  id: label  
  anchors.horizontalCenter: parent.horizontalCenter  
  plainText: "Email: name@example.com"  
}
```

```
LinkedList {  
  anchors.horizontalCenter: parent.horizontalCenter  
  color: Theme.primaryColor  
  linkColor: Theme.highlightColor  
  plainText: "Company Ltd - example.com"  
}
```

Email: [name@example.com](#)

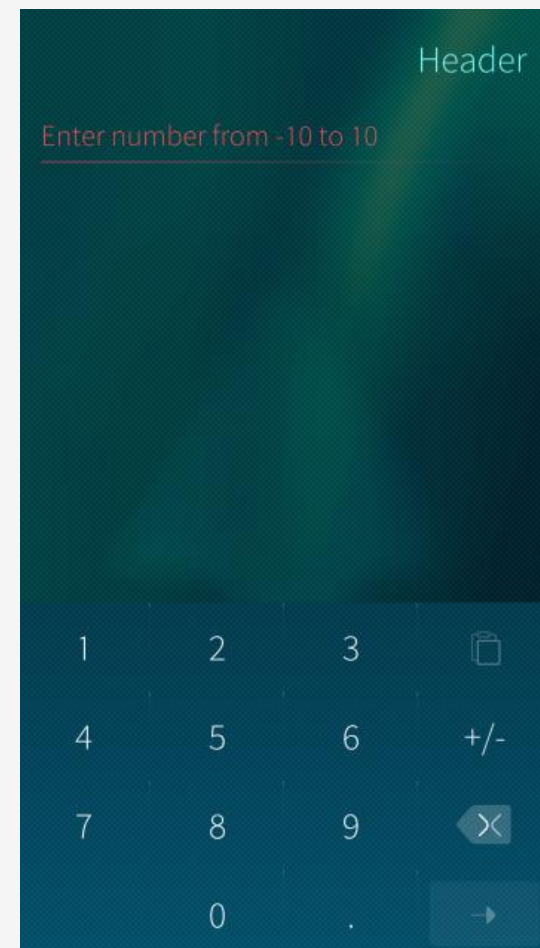
Company Ltd - [example.com](#)

TextField – ввод однострочного текста

- **text** : **string** — вводимый текст
- **placeholderText** : **string** — текст, отображаемый до ввода
- **label** : **string** — текст, отображаемый под полем ввода
- **readOnly** : **bool** — запрет ввода текста
- **inputMethodHints** : **Qt::InputMethodHints** — режим клавиатуры для ввода
- **validator** : **Validator** — проверка ввода
- **copy()**, **cut()**, **paste()** — копировать, вырезать, вставить текст
- **select()**, **selectAll()**, **selectWord()** — выбрать текст

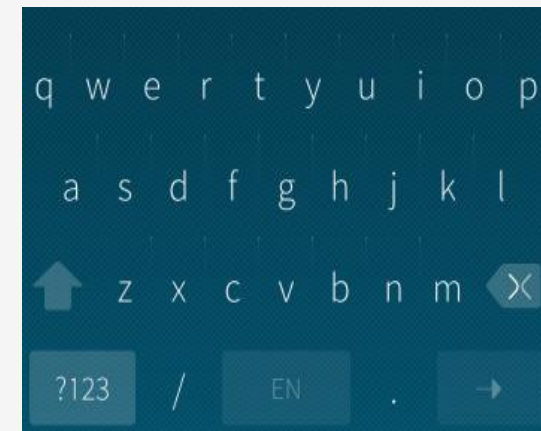
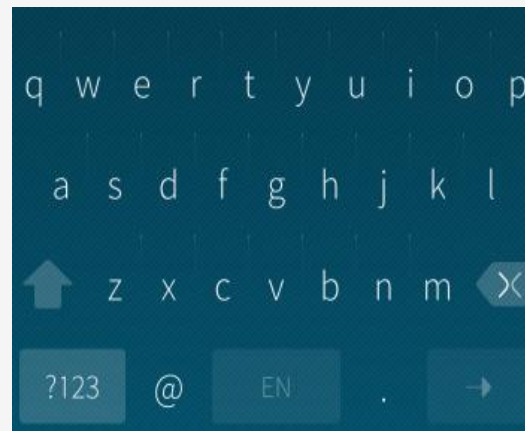
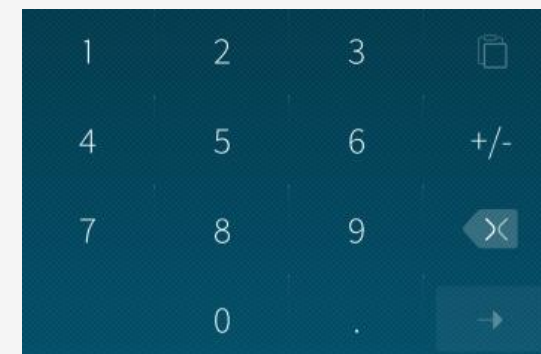
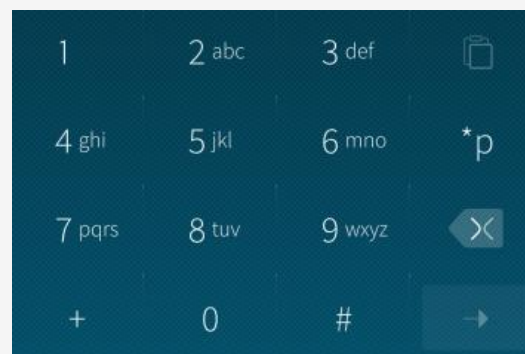
TextField – однострочный текст

```
TextField {  
    width: parent.width  
    placeholderText: qstr("Enter number from -10 to 10")  
    label: qstr("Text Field")  
    inputMethodHints: Qt.ImhFormattedNumbersOnly  
    validator: DoubleValidator {  
        bottom: -10; top: 10  
        decimals: 2  
    }  
    EnterKey.enabled: !errorHighlight  
    EnterKey.iconSource: "image://theme/icon-m-enter-next"  
    EnterKey.onClicked: console.log(text)  
}
```



InputMethodHint – управление вводом

- **Qt.ImhDialableCharactersOnly** — номер телефона
- **Qt.ImhDigitsOnly** — целые числа
- **Qt.ImhEmailCharactersOnly** – E-mail
- **Qt.ImhFormattedNumbersOnly** – действительные числа
- **Qt.ImhUrlCharactersOnly** – WEB-адрес
- **Qt.ImhNoPredictiveText** – не использовать подсказки
- **Qt.ImhNoAutoUppercase** – не переключаться автоматически на верхний регистр



Validator – проверка ввода

IntValidator — проверка целых чисел

- `bottom, top : int` — границы интервала
- `locale : string` — название локали

DoubleValidator — проверка чисел с плавающей точкой

- `bottom, top : real` — границы интервала
- `locale : string` — название локали
- `decimals : int` — количество цифр после десятичной точки
- `notation : enumeration` — представление числа
 - `DoubleValidator.ScientificNotation` — научное (по умолчанию)
 - `DoubleValidator.StandardNotation` — обычное

RegexValidator — проверка регулярных выражений

- `regExp : regExp` — регулярное выражение

Validator – проверка ввода

```
TextField {  
    placeholderText: qsTr("Input login...")  
    label: qsTr("Login")  
    width: parent.width  
    validator: RegExpValidator {  
        regexp: /^[a-z][a-z0-9_-]{2,15}$/  
    }  
    inputMethodHints: Qt.ImhNoAutoUppercase  
}
```



TextArea – многострочный ввод

horizontalAlignment : **enumeration** — горизонтальное выравнивание текста

verticalAlignment : **enumeration** — вертикальное выравнивание текста

wrapMode : **enumeration** — режим переноса текста

- `TextEdit.NoWrap` — нет переноса
- `TextEdit.WordWrap` — перенос на границу слова
- `TextEdit.WrapAnywhere` — перенос в любой точке строки
- `TextEdit.Wrap` — перенос на границе слова по возможности

TextArea – многострочный ввод

```
TextArea {  
    width: parent.width  
    label: qsTr("Multi-line text")  
    placeholderText: qsTr("Input something and press Enter")  
    wrapMode: TextEdit.WordWrap  
}
```

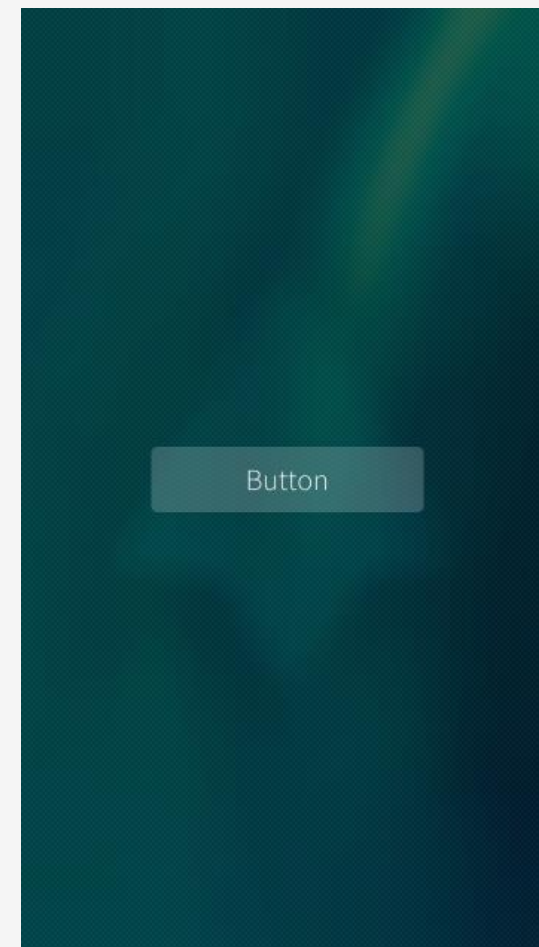
```
TextArea {  
    width: parent.width  
    label: qsTr("Multi-line text")  
    text: "String 1\nString 2\nString 3"  
}
```



Button - кнопка

- `preferredWidth : real` — размер кнопки
 - `Theme.buttonWidthExtraSmall`
 - `Theme.buttonWidthMedium`
 - `Theme.buttonWidthLarge`
- `text : string` — отображаемый текст на кнопке

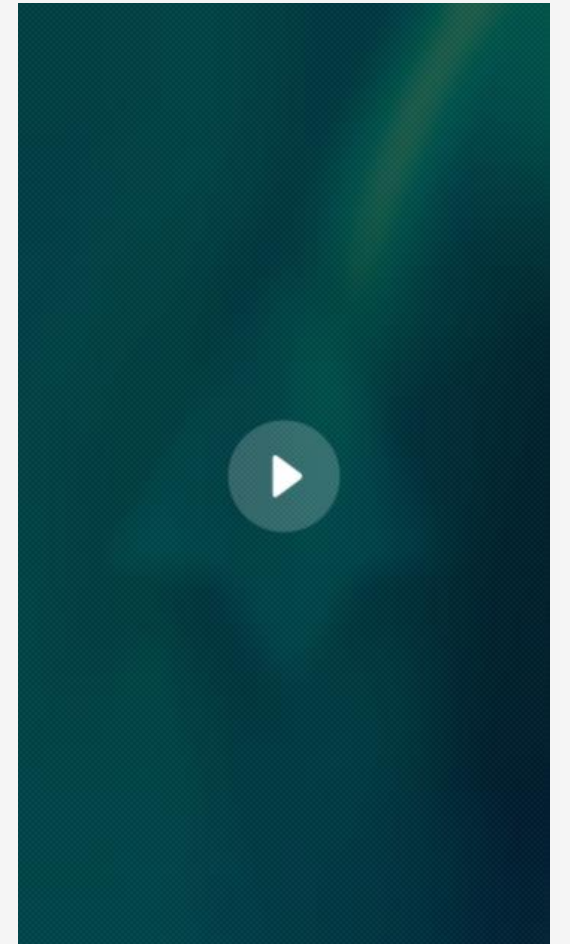
```
Page {  
  Button {  
    anchors.centerIn: parent  
    preferredWidth : Theme.buttonWidthMedium  
    text: "Button"  
    onClicked: console.log("Button clicked")  
  }  
}
```



IconButton – кнопка в виде иконки

- `icon : Image` — изображение для кнопки
- `highlighted : bool` — подсвечена ли кнопка

```
IconButton {  
  anchors.centerIn: parent  
  icon.source: "image://theme/icon-m-play"  
  icon.scale: 2  
  onClicked: console.log("Button pressed")  
}
```



ValueButton – кнопка со значением

- `label : string` — поле для краткого описания
- `value : string` — поле для значения
- `description : string` — поле для комментария

```
ValueButton {  
    property int count: 0  
  
    label: qstr("Button")  
    description: qstr("Counting Value")  
    value: count  
    onClicked: count++  
}
```



Button 0
Counting Value

Switch – кнопка переключения

Switch унаследован от компонента **MouseArea**

- **automaticCheck** : **bool** — автоматическое переключение при нажатии
- **checked** : **bool** — статус переключателя
- **busy** : **bool** — находится ли переключатель в состоянии «занят»
- **icon** : **Image** — иконка для переключателя

TextSwitch — переключатель с полем для текста

- **text** : **string** — текст рядом с индикатором статуса
- **description** : **string** — описание переключателя

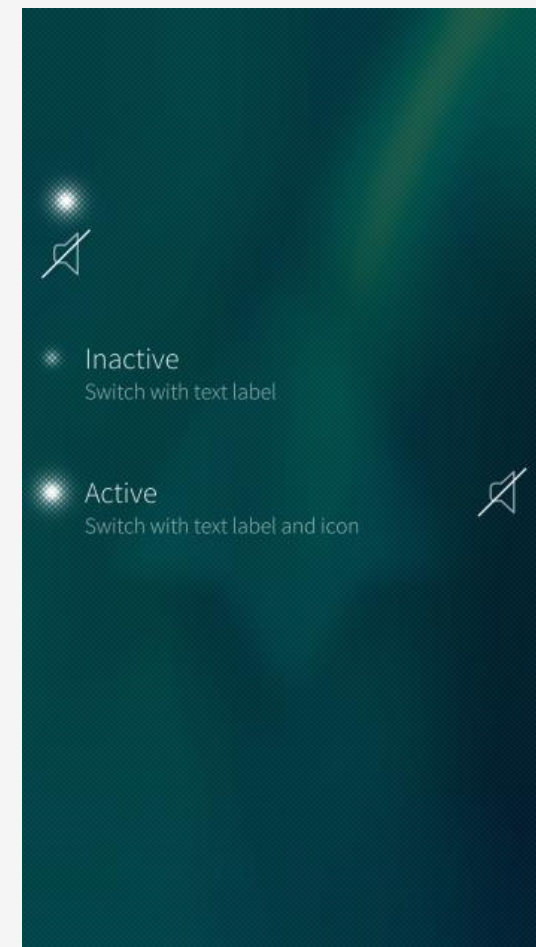
IconTextSwitch — переключатель с иконкой и полем для текста

Примеры переключателей

```
Switch {  
    icon.source: "image://theme/icon-m-speaker-  
mute"  
}
```

```
TextSwitch {  
    text: checked ? qsTr("Active") : qsTr("Inactive")  
    description: qsTr("Switch with text label")  
}
```

```
IconTextSwitch {  
    icon.source: "image://theme/icon-m-speaker-  
mute"  
    text: checked ? qsTr("Active") : qsTr("Inactive")  
    description: qsTr("Switch with text label and icon")  
}
```



Keypad – экранная клавиатура

- `symbolsVisible` : `bool` — true для отображения "*" и "#"
- `vanityDialNumbersVisible` : `bool` — true для вывода символов под цифрами
- `voiceMailIconSource` : `url` — адрес иконки для голосовой почты
- `onClicked()`, `onPressed()`, `onReleased()`, `onCanceled()` — вызов обработчика соответствующего сигнала

```
Keypad{  
    anchors.bottom: parent.bottom  
}
```



TouchBlocker – перехват нажатий

Ограничивает взаимодействие с
элементами на странице

```
SilicaListView {  
    id: slv  
    header: PageHeader { title: qsTr("List Page") }  
    anchors.fill: parent  
}  
TouchBlocker { anchors.fill: slv }
```



Контейнеры

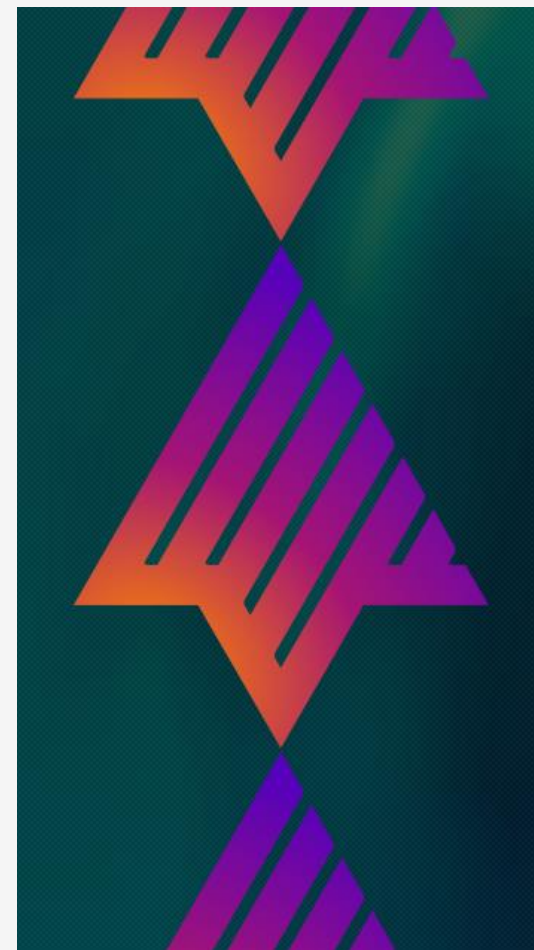
- [SilicaFlickable](#) — наследник [Flickable](#)
- [SilicaListView](#) — список, наследник [ListView](#)
- [SilicaGridView](#) — таблица, наследник [GridView](#)
- [SlideshowView](#) — циклический показ слайдов, наследник [PathView](#)
- [SilicaWebView](#) — WEB-страница, наследник [WebView](#)

SilicaFlickable – общий контейнер

- **pullDownMenu, pushUpMenu** : **Item** — добавляют вытягиваемые меню
- **quickScroll** : **bool** — включена ли вертикальная область быстрой прокрутки
- **scrollToBottom()** — быстрая прокрутка вниз
- **scrollToTop()** — быстрая прокрутка вверх
- **ScrollDecorator** — ползунок прокрутки
 - **HorizontalScrollDecorator** — горизонтальный
 - **VerticalScrollDecorator** — вертикальный

SilicaFlickable – общий контейнер

```
SilicaFlickable {  
    id: flickable  
    contentWidth: column.width  
    contentHeight: column.height  
  
    Column {  
        id: column  
        width: flickable.width  
        PageHeader { title: qsTr("Flickable Page") }  
        Image {  
            source: "logo_aurora.svg"  
            width: sourceSize.width  
            height: flickable.height * 1.5  
            anchors.horizontalCenter: parent.horizontalCenter  
            fillMode: Image.TileVertically  
        }  
    }  
    VerticalScrollDecorator {}  
}
```

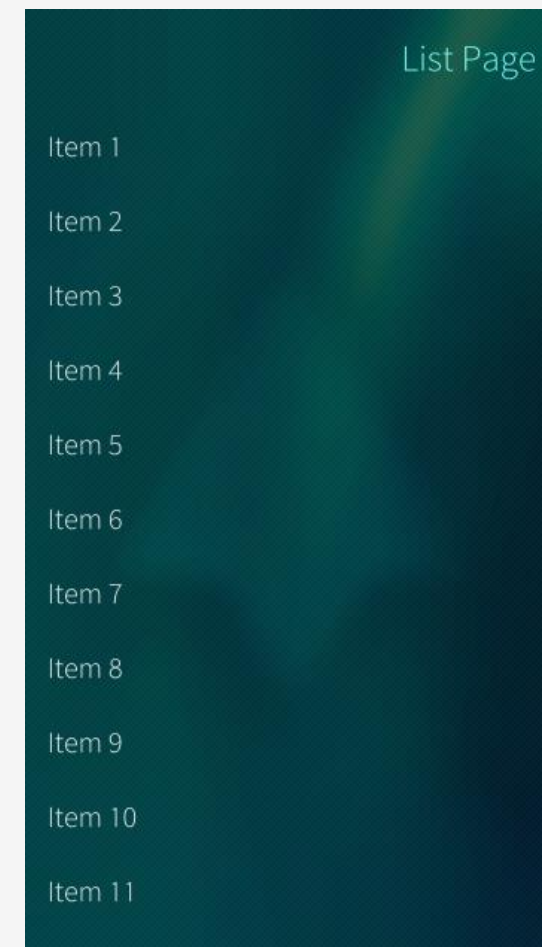


SilicaListView – Список элементов

- **model** — данные, которые будут отображаться в списке
- **delegate** — как именно будут отображаться данные в списке
- **quickScroll**: **bool** — включена ли вертикальная область быстрой прокрутки
- **VerticalScrollDecorator** — вертикальный ползунок прокрутки

SilicaListView – Список элементов

```
SilicaListView {  
  anchors.fill: parent  
  header: PageHeader { title: qsTr("List Page") }  
  delegate: ListItem {  
    Label {  
      text: qsTr("Item %1").arg(model.index + 1)  
      anchors.verticalCenter: parent.verticalCenter  
      x: Theme.horizontalPageMargin  
      color: highlighted ?  
        Theme.highlightColor :  
        Theme.primaryColor  
    }  
  }  
  model: 100  
  
  VerticalScrollDecorator {}  
}
```

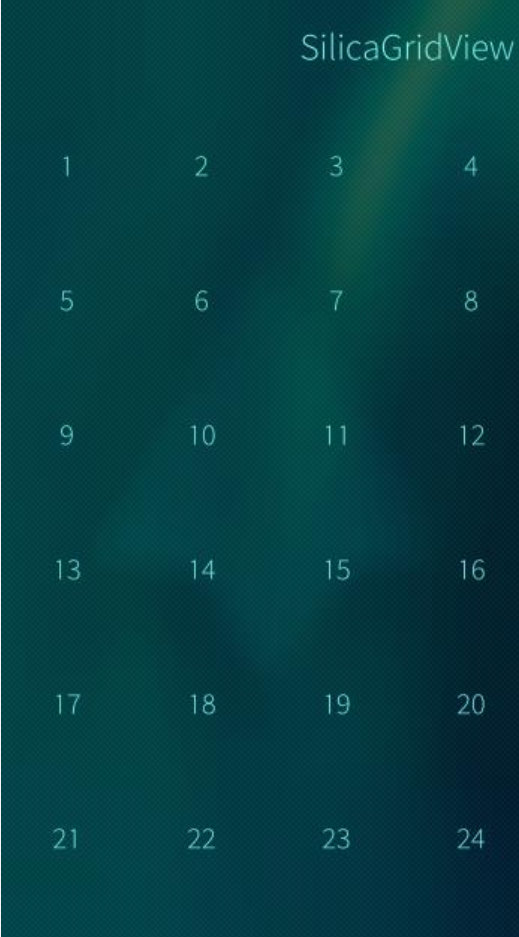


SilicaGridView – Таблица

- **cellHeight** – высота ячеек в сетке
- **cellWidth** – ширина ячеек в сетке
- **quickScroll**: **bool** — включена ли вертикальная область быстрой прокрутки
- **ScrollDecorator** — ползунок прокрутки
 - **HorizontalScrollDecorator** — горизонтальный
 - **VerticalScrollDecorator** — вертикальный

SilicaGridView – Таблица

```
SilicaGridView {  
    anchors.fill: parent  
    header: PageHeader { title: qsTr("SilicaGridView") }  
    cellWidth: width / 4  
    cellHeight: cellWidth  
    delegate: Label {  
        width: GridView.view.cellWidth  
        height: GridView.view.cellHeight  
        text: (model.index + 1)  
        color: Theme.highlightColor  
        verticalAlignment: Text.AlignVCenter  
        horizontalAlignment: Text.AlignHCenter  
    }  
    model: 100  
  
    VerticalScrollDecorator {}  
}
```



The screenshot shows a dark-themed application window titled "SilicaGridView". It displays a grid of 24 numbers arranged in 6 rows and 4 columns. The numbers are 1 through 24, starting from the top-left cell and increasing sequentially to the bottom-right cell. The grid is styled with a dark background and light-colored text.

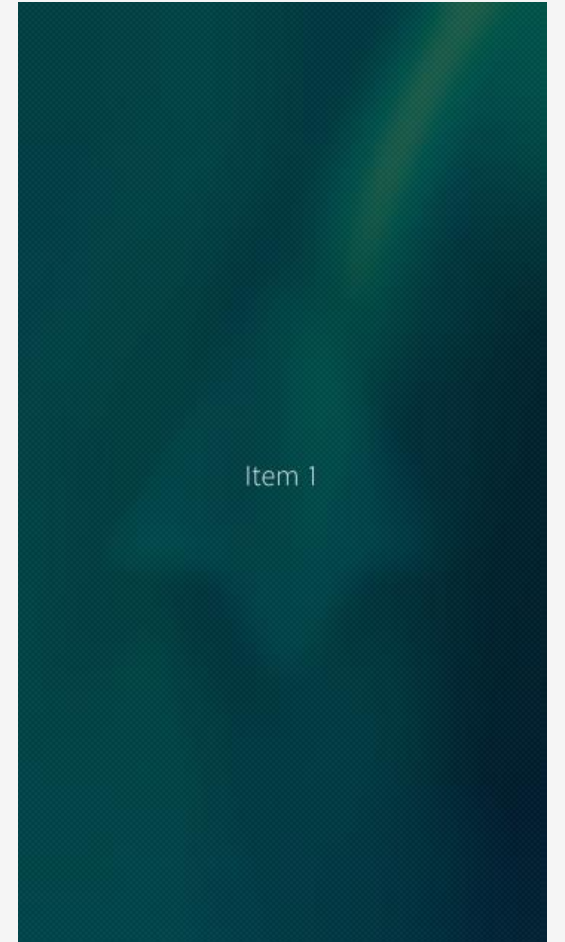
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24

SlideShowView – слайды по циклу

- **itemHeight** : **real** — высота каждого элемента в типе
- **itemWidth** : **real** — ширина каждого элемента в типе
- **orientation** : **enumeration** — направление переключения слайдов
 - **Qt.Horizontal** — горизонтально
 - **Qt.Vertical** — вертикально

SlideShowView – слайды по циклу

```
SlideshowView {  
    id: slideshowView  
    anchors.fill: parent  
    delegate: BackgroundItem {  
        width: slideshowView.itemWidth  
        height: slideshowView.itemHeight  
  
        Label {  
            text: qsTr("Item %1").arg(index + 1)  
            anchors.centerIn: parent  
            color: highlighted ?  
                Theme.highlightColor :  
                Theme.primaryColor  
        }  
    }  
    model: 5  
}
```



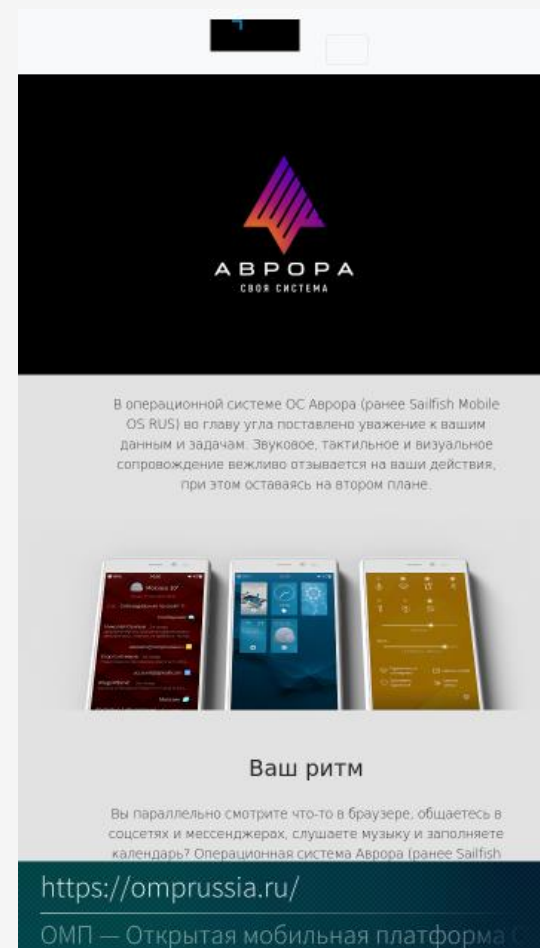
SilicaWebView – веб-страница

- **url** : **string** — адрес веб-страницы
- **title** : **string** — заголовок загружаемой страницы (только для чтения)
- **loading** : **bool** — идет ли процесс загрузки страницы (только для чтения)
- **loadProgress** : **real** — прогресс загрузки страницы (только для чтения)
- **quickScroll** : **bool** — включена ли вертикальная область быстрой прокрутки

SilicaWebView – веб-страница

```
SilicaWebView {  
  id: webView  
  anchors { fill: parent; bottomMargin: urlField.height }  
  url: "https://omprussia.ru"  
}
```

```
TextField {  
  id: urlField  
  anchors {  
    left: parent.left; right: parent.right  
    bottom: parent.bottom  
  }  
  text: webView.url  
  label: webView.title  
  EnterKey.onClicked: webView.url = text  
}
```



MenuItem и MenuLabel

MenuItem — интерактивный элемент меню

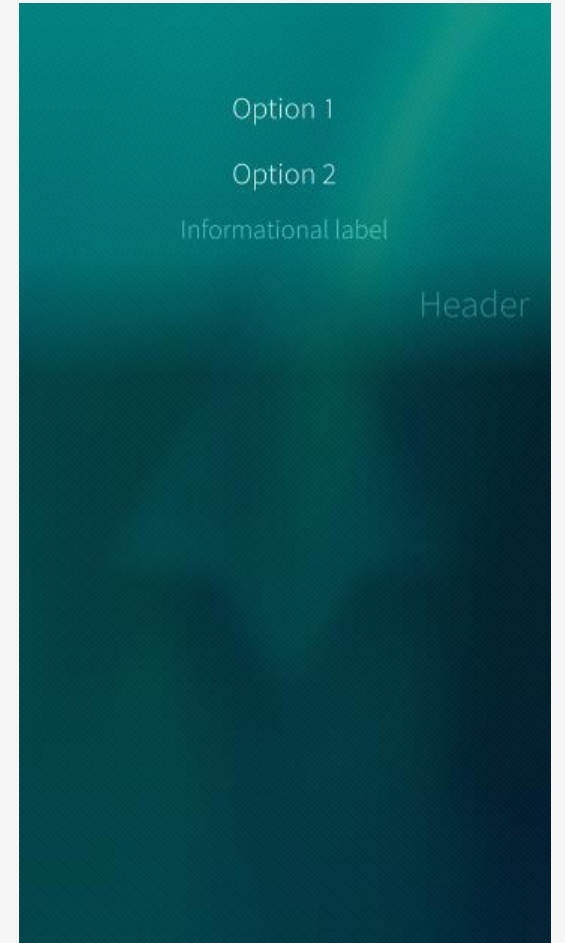
- **text** – текст пункта меню
- **color** – цвет текст пункта меню
- **font** – группа свойств для настройки шрифта
- **horizontalAlignment** – горизонтальное выравнивание текста

MenuLabel — текст в меню

- **text** – текст метки в меню
- **color** – цвет текста метки в меню
- **verticalOffset** – вертикальный отступ

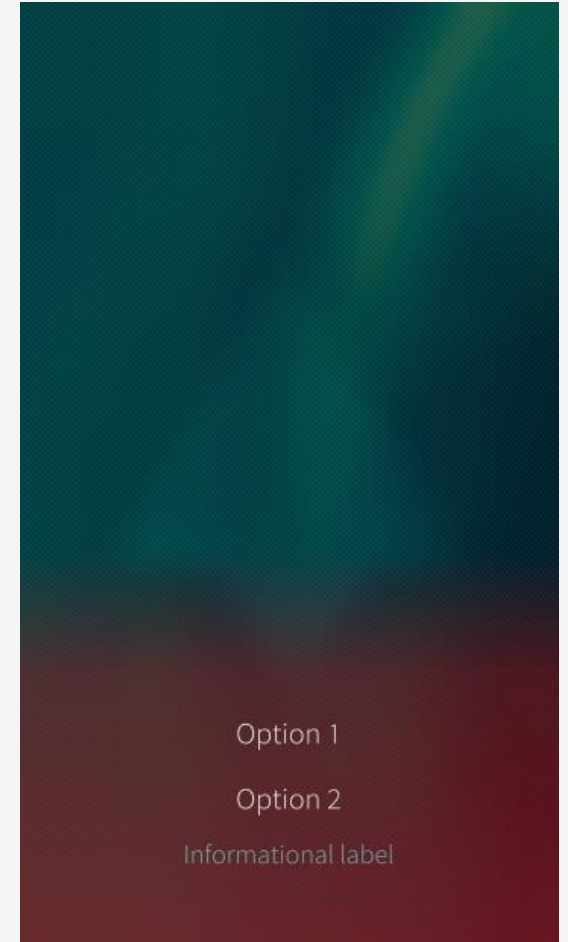
PullDownMenu – верхнее меню

```
PullDownMenu {  
    MenuItem {  
        text: qsTr("Option 1")  
        onClicked: console.log(qsTr("Option"))  
    }  
    MenuItem {  
        text: qsTr("Option 2")  
        onClicked: console.log(qsTr("Option"))  
    }  
    MenuLabel {  
        text: qsTr("Informational label")  
    }  
}
```



PushUpMenu – нижнее меню

```
PushUpMenu {  
    backgroundColor: "red"  
    highlightColor: backgroundColor  
  
    MenuItem {  
        text: qsTr("Option 1")  
        onClicked: console.log(qsTr("Option"))  
    }  
    MenuItem {  
        text: qsTr("Option 2")  
        onClicked: console.log(qsTr("Option"))  
    }  
    MenuLabel {  
        text: qsTr("Informational label")  
    }  
}
```



Listltem и ContextMenu – элемент списка с МЕНЮ

Listltem

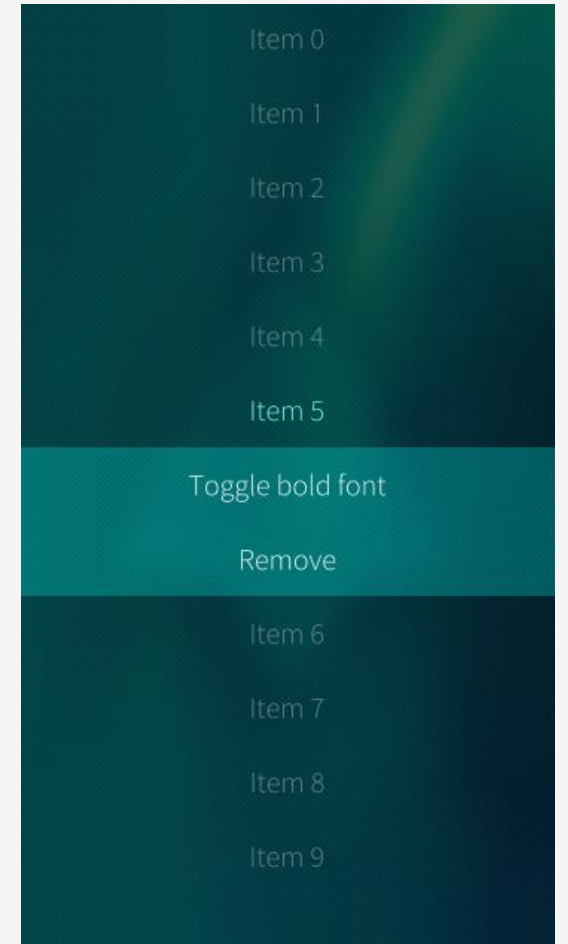
- `contentHeight` : `real` — высота без меню
- `highlighted` : `bool` — открыто ли меню
- [read-only] `menuOpen` : `bool` — true, если меню открыто
- `openMenuOnPressAndHold` : `bool` — true для открытия меню при удержании
- `closeMenu()` — скрыть меню
- `openMenu()` — отобразить меню
- `removeAction()` — устанавливает действие обратного вызова

ContextMenu

- `active` : `bool` — отображается ли меню в данный момент
- `closeOnActivation` : `bool` — закрыть ли меню после нажатия на MenuItem
- `onActivated(int index)` — вызвать обработчик сигнала при нажатии на MenuItem
- `open()`, `close()` — открыть/закрыть контекстное меню

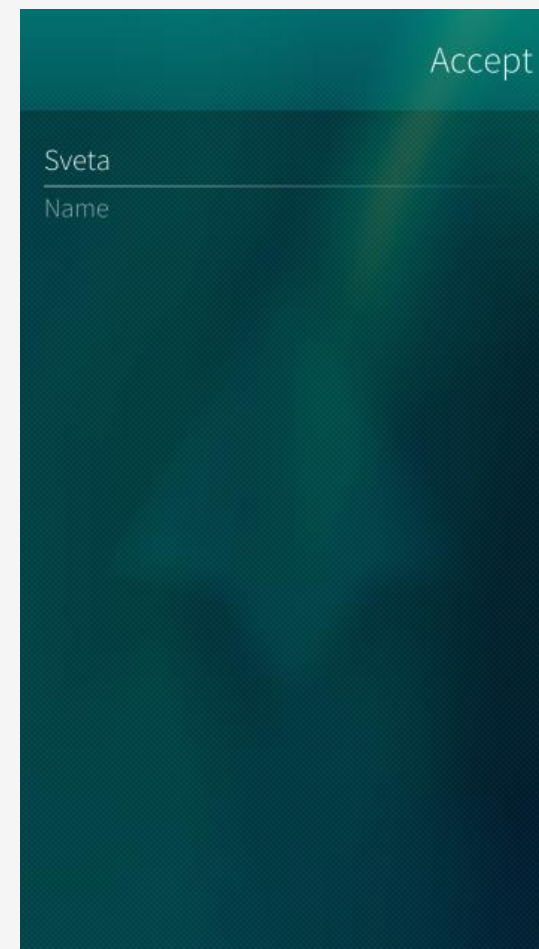
ListItem и ContextMenu

```
ListItem {  
    menu: ContextMenu {  
        MenuItem {  
            text: qsTr("Toggle bold font")  
            onClicked: label.font.bold = !label.font.bold  
        }  
    }  
}  
  
Label {  
    id: label  
    text: qsTr("Item %1").arg(model.index + 1)  
    anchors.centerIn: parent  
}  
}
```



Dialog – страница подтверждения

- **canAccept** : **bool** — можно ли подтвердить диалог
- **result** : **enumeration** — результат закрытия диалога
- **open()** — открыть диалог
- **accept()** — подтвердить действия
- **reject()** — отменить диалог
- **close()** — закрыть диалог
- **onOpened()** — обработчик сигнала при открытии
- **onAccepted()** — обработчик сигнала при подтверждении
- **onRejected()** — обработчик сигнала при отмене
- **onDone()** — обработчик сигнала перед закрытием диалога до вызова **onAccepted()** и **onRejected()**



DialogHeader – заголовок диалога

- **acceptText** : **string** — текст для действия подтверждения диалога
- **cancelText** : **string** — текст для действия отмены диалога
- **dialog** : **Item** — указывает на диалог, к которому относится заголовок
- **title** : **string** — текст заголовка
- **extraContent** : **Item** — элемент для вставки дополнительного контекста

Пример диалога

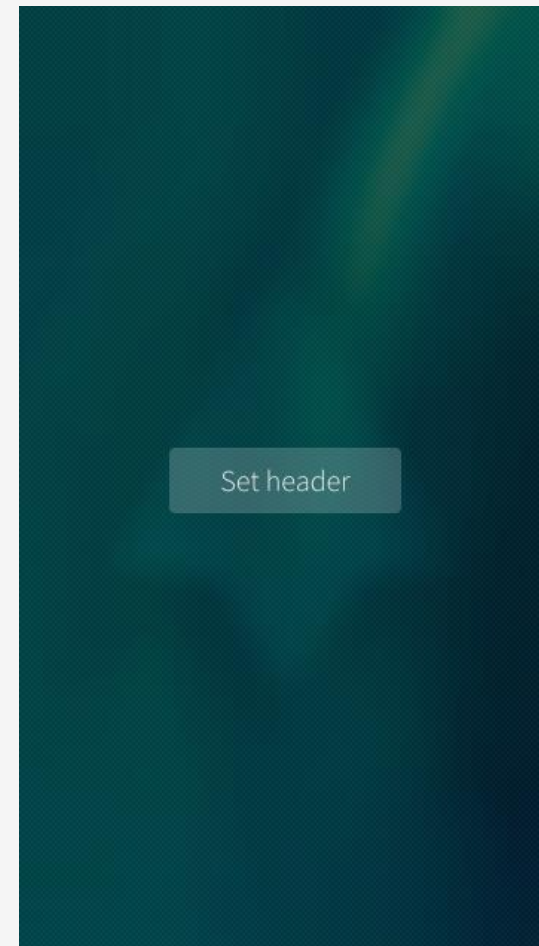
```
Dialog {  
    property string name  
  
    DialogHeader {  
        acceptText: qsTr("Set header")  
        title: name  
    }  
    TextField {  
        id: nameField  
        anchors.centerIn: parent  
        width: parent.width  
        placeholderText: qsTr("Enter page name")  
        label: qsTr("Page name")  
    }  
    onDone: if (result == DialogResult.Accepted)  
        name = nameField.text  
}
```



Вызов диалога со страницы

```
PageHeader { id: header }

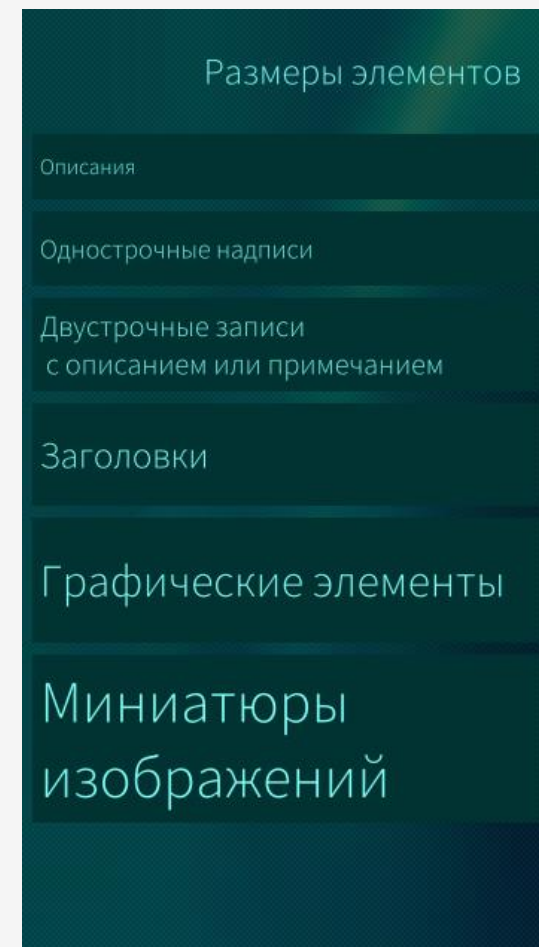
Button {
    text: qsTr("Set header")
    anchors.centerIn: parent
    onClicked: {
        var dialog = pageStack.push(
            Qt.resolvedUrl("HeaderInputDialog.qml"),
            {"name": header.title}
        );
        dialog.accepted.connect(function() {
            header.title = dialog.name;
        });
    }
}
```



Размеры элементов в Silica

Присваивается свойствам размера объекта

- `Theme.itemSizeExtraSmall`
- `Theme.itemSizeSmall`
- `Theme.itemSizeMedium`
- `Theme.itemSizeLarge`
- `Theme.itemSizeExtraLarge`
- `Theme.itemSizeHuge`



Размеры шрифтов в Silica

Для свойства `font.pixelSize`

- `Theme.fontSizeTiny`
- `Theme.fontSizeExtraSmall`
- `Theme.fontSizeSmall`
- `Theme.fontSizeMedium`
- `Theme.fontSizeLarge`
- `Theme.fontSizeExtraLarge`
- `Theme.fontSizeHuge`

Размеры шрифтов

ExtraSmall: для очень маленьких пространств, следует избегать

Small: второстепенная информация, дополнительные описания, которые можно опустить

Medium: стандартный текст для списков и абзацев

Large: текст для заголовков

ExtraLarge:
полноэкранные
надписи и уведомления

Huge: текст для
объявлений

Цвета в Silica

- Для свойства **color**

`Theme.primaryColor`

`Theme.secondaryColor`

`Theme.highlightColor`

`Theme.secondaryHighlightColor`

`Theme.highlightBackgroundColor`

`Theme.highlightDimmerColor`

- Для прозрачного фона

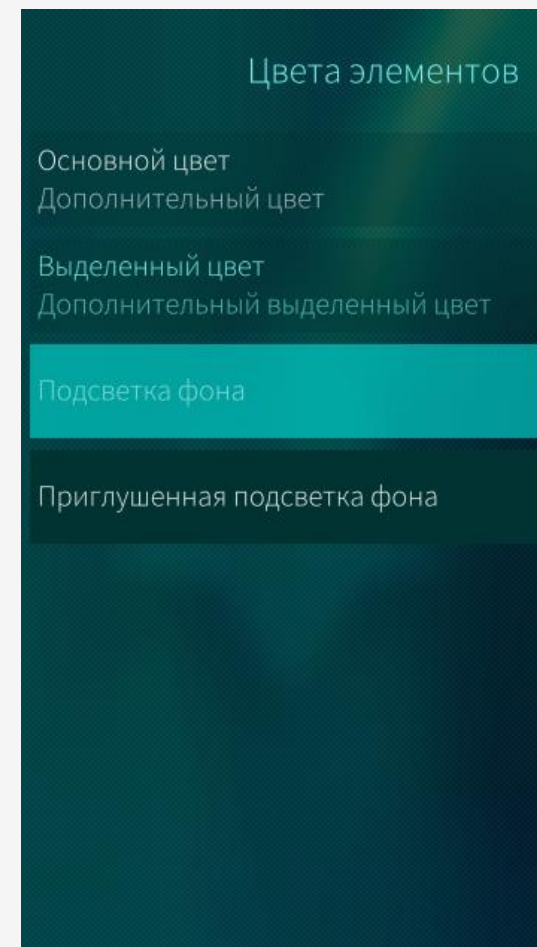
`Theme.rgb(Theme.highlightDimmerColor,`

`Theme.highlightBackgroundOpacity)`

`Theme.colorScheme` — цветовая схема

`Theme.LightOnDark`

`Theme.DarkOnLight`





Отступы в Silica





















Theme.horizontalPageMargin — отступы от границ экрана

Theme.paddingLarge — отступы между элементами

Theme.paddingMedium — отступы между встроенными
элементами

Theme.paddingSmall — между метками и иными мелкими
элементами

Иконки действий

Icon	Name	Icon	Name	Icon	Name
	icon-cover-alarm		icon-cover-answer		icon-cover-backup
	icon-cover-camera		icon-cover-cancel		icon-cover-dialer
	icon-cover-favorite		icon-cover-hangup		icon-cover-location
	icon-cover-message		icon-cover-mute		icon-cover-new
	icon-cover-next-song		icon-cover-next		icon-cover-pause
	icon-cover-people		icon-cover-play		icon-cover-previous-song
	icon-cover-previous		icon-cover-refresh		icon-cover-reject
	icon-cover-search		icon-cover-shuffle		icon-cover-subview
	icon-cover-sync		icon-cover-timer		icon-cover-transfers
	icon-cover-unmute				

Размеры иконок

- **CoverAction** — иконки для обложек
- **Theme.iconSizeExtraSmall** — самый маленький размер
- **Theme.iconSizeSmall** — маленькие
- **Theme.iconSizeSmallPlus** — чуть больший вариант iconSizeSmall
- **Theme.iconSizeMedium** — средние
- **Theme.iconSizeLarge** — большие
- **Theme.iconSizeExtraLarge** — самый большой размер значков
- **Theme.iconSizeLauncher** — размер, который используется для значков на домашнем экране

Масштабирование элементов

Масштабирование элементов интерфейса
в зависимости от размера экрана
с помощью `Theme.pixelRatio` или `Theme.dp`

```
Rectangle {  
  color: "red"  
  anchors.centerIn: parent  
  // width: 100 * Theme.pixelRatio  
  width: Theme.dp(100)  
  height: Theme.itemSizeSmall  
}
```

Screen.sizeCategory

– размеры экрана

Screen.sizeCategory

- Screen.Small
- Screen.Medium
- Screen.Large
- Screen.ExtraLarge

```
Label {
    text: "Hello world!"
    anchors.centerIn: parent
    font.pixelSize:
        Screen.sizeCategory >= Screen.Large
        ? Theme.fontSizeLarge
        : Theme.fontSizeNormal
}

ApplicationWindow {
    InitialPage:
        Screen.sizeCategory >= Screen.Large
        ? Qt.resolvedUrl("SplitViewPage.qml")
        : Qt.resolvedUrl("ListViewPage.qml")
}
```

Palette – цветовая палитра

- **colorScheme** : **enumeration** — цветовая схема палитры
- **errorColor** : **color** — используется для обозначения ошибок
- **highlightBackgroundColor** : **color** — фон выделенного текста
- **highlightColor** : **color** — основной цвет для неинтерактивного текста
- **primaryColor** : **color** — цвет активных элементов
- **secondaryColor** : **color** — цвет для менее значимых элементов
- Свойства **Palette** соответствуют одноимённым в **Theme**

Модуль Silica Background

- Настройка фона приложения
- Типы модуля:
 - **Background** — отдельные представления на большом общем полноэкранном фоне
 - **Wallpaper** — фоновое изображение, подходящее для прямого отображения или использования в качестве поставщика текстуры
- Модуль находится в стадии разработки и может быть изменён
- Строка импорта:

```
import Sailfish.Silica.Background 1.0
```

Background

Независимый от позиции фоновый элемент

- **material** : **Material** — материал, содержащий шейдеры GLSL
- **sourceItem** : **Item** — элемент, который может отображаться в полноэкранном режиме
- **patternItem** : **Item** — элемент, который можно дублировать тайлами
- **fillMode** : **enumeration** — масштабирование элемента фона
 - **Background.Strech**
 - **Background.PreserveAspectWidth**
 - **Background.PreserveAspectSquare**

Стандартные варианты фона

- **ColorBackground** — простой материал с цветом
 - **color** — цвет фона
- **ThemeBackground** — элемент фона, который отображается с использованием материала темы
 - **backgroundMaterial** : **string** — фоновый материал темы для отображения обоев и шаблона glass
 - **color** — накладываемый цвет
 - **highlightColor** — цвет подсветки

Material – материал шейдера

- `vertexShader` — вершинный шейдер
- `fragmentShader` — фрагментный шейдер

```
uniform lowp sampler2D sourceTexture;
```

```
uniform lowp sampler2D patternTexture;
```

```
varying highp vec2 sourceCoord;
```

```
varying highp vec2 patternCoord;
```

```
void backgroundMain() {
```

```
    lowp vec4 pattern = texture2D(patternTexture, patternCoord) * 0.1;
```

```
    gl_FragColor = texture2D(sourceTexture, sourceCoord);
```

```
    gl_FragColor = (gl_FragColor * (1.0 - pattern.a)) + pattern;
```

```
}
```



Типы материалов

- **Синглтон *Materials*** — набор стандартных разделяемых материалов *Background*
- ***BlurMaterial*** — показывает изображение и накладывает цвет
- ***GlassMaterial*** — фоновый рисунок, узор и наложенный цвет в стиле стеклянной темы

Типы Wallpaper

- **FilteredImage** позволяет применить серию эффектов к изображению
 - **filters** — список фильтров
 - **sourceItem** — источник для отфильтрованного изображения
- **ImageWallpaper** — квадратное обрезанное изображение
- **ThemeWallpaper** — изображение обоев, к которому применен фильтр темы
 - **explicitFilters** — список фильтров
 - **wallpaperFilter** — фильтр для обоев темы glassBlurur
- **ThemeImageWallpaper** — квадратное обрезанное и обработанное фильтром темы изображение
 - **imageUrl** — URL источника

Фильтры для FilteredImage

- **Синглтон Filters** — набор стандартных разделяемых фильтров
- **ConvolutionFilter** — применяет ядро свёртки к **FilteredImage**
- **GlassBlur** — фильтр, который размывает изображение для создания эффекта стеклянной темы
- **RepeatFilter** — повторяет последовательность фильтров изображения фиксированное количество раз
- **ResizeFilter** — изменяет размер вывода предыдущего шага фильтра
- **SequenceFilter** — собирает последовательность фильтров изображений в одно определение фильтра
- **ShaderFilter** — фильтрует изображение с помощью программы шейдера GLSL



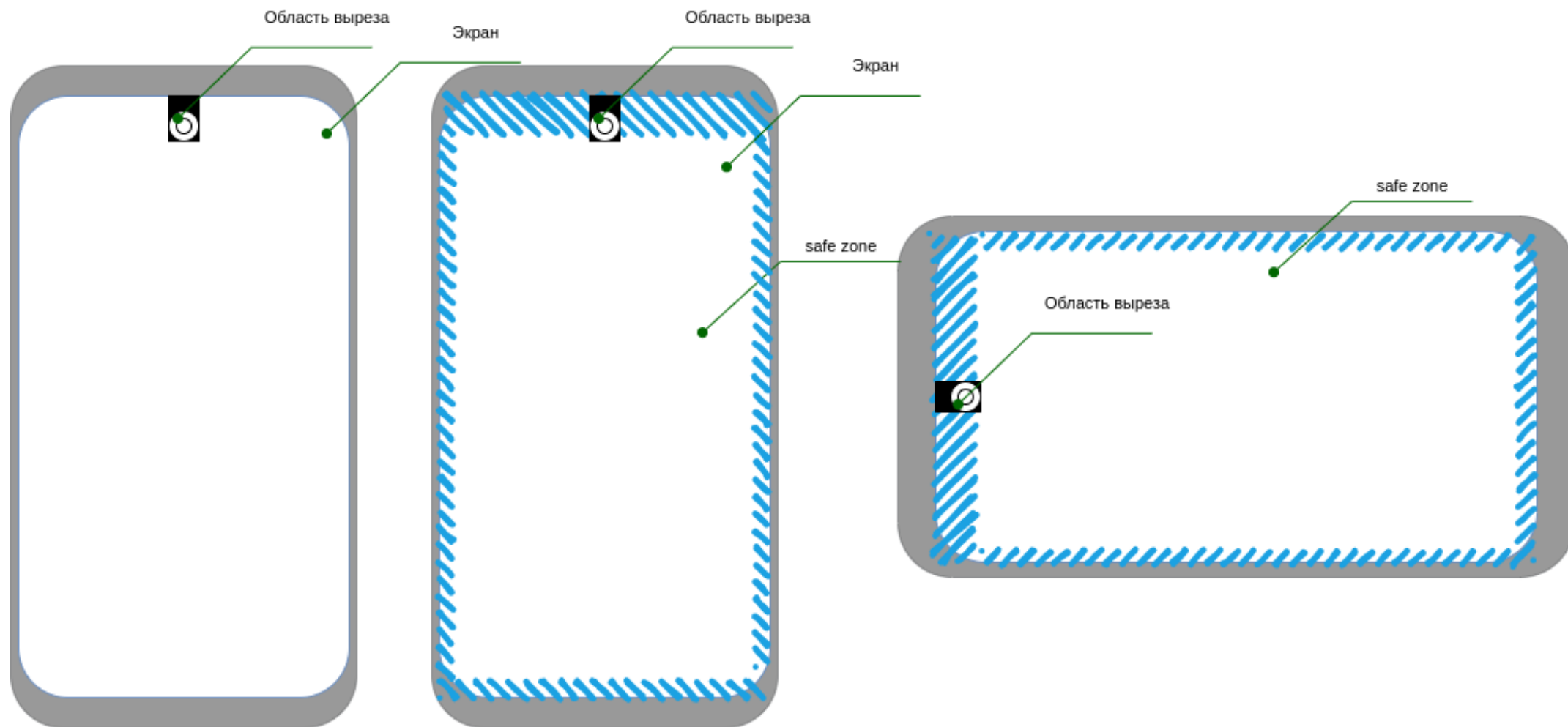
Работа с вырезами

Системный вырез – обеспечивает место для камеры или датчиков на передней панели устройства

Варианты работы с вырезами:

- Выбрать режим отображения приложения
- Напрямую получать значения безопасной зоны и компонентов, с которыми возможна коллизия

SafeZone – безопасная зона



Выбор режима отображения

Настройка `displayMode` у `Application Window`

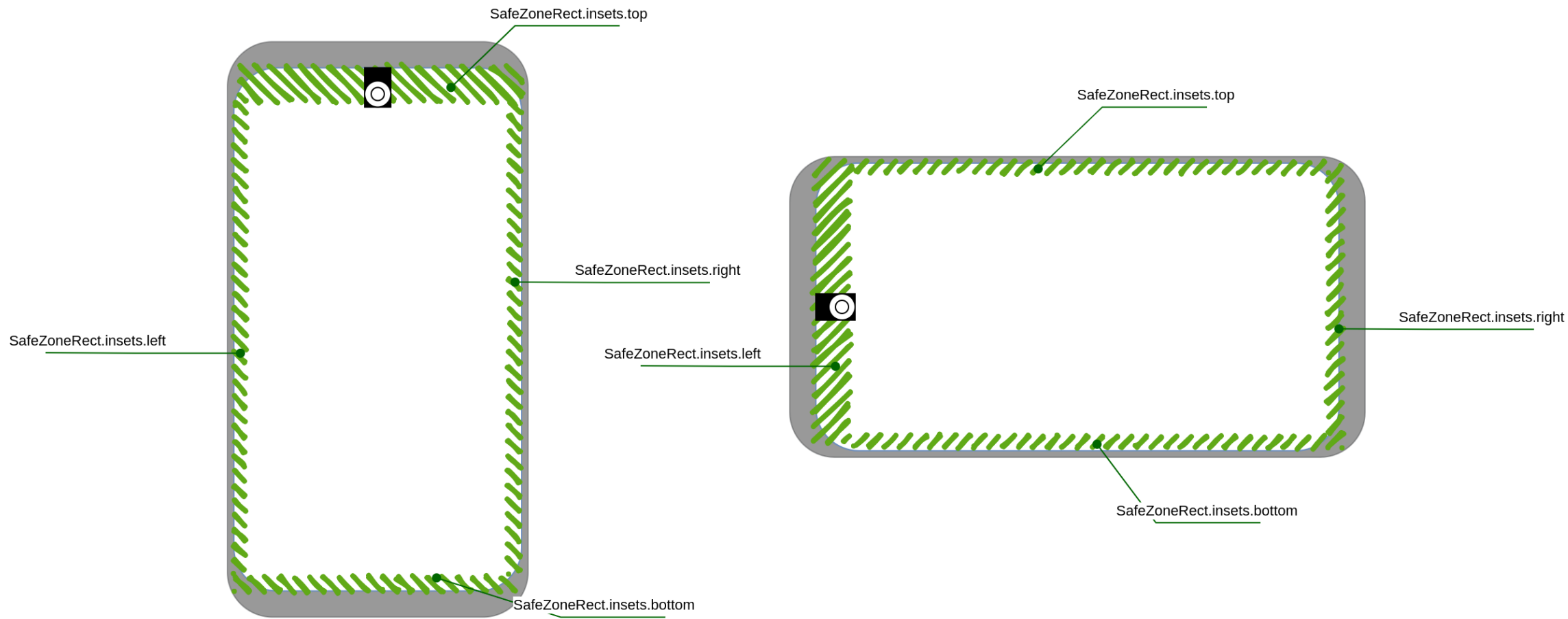
`ApplicationDisplayMode.FullPortrait`

`ApplicationDisplayMode.FillScreen`

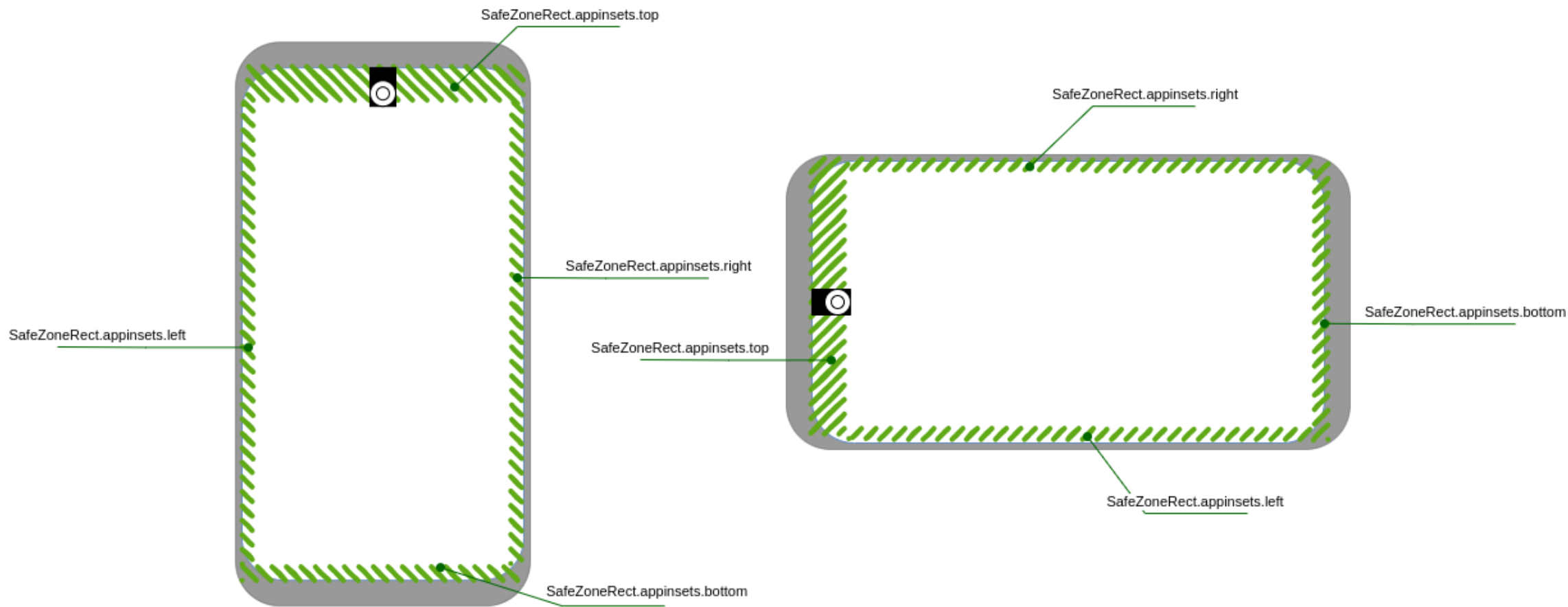
`ApplicationDisplayMode.SafeZone`

```
ApplicationWindow {  
...  
  Component.onCompleted: {  
    if (applicationWindow.hasOwnProperty("displayMode")) {  
      applicationWindow.displayMode =  
        ApplicationDisplayMode.FullPortrait  
    }  
  }  
}
```

Вырезы: SafeZoneRect.insets



Вырезы: SafeZoneRect.appinsets



Взаимодействие со строкой состояния

Свойства `ApplicationWindow`

`statusbarForceVisible` —

принудительно включить/отключить отображение строки состояния, если


установлен `displayMode FullPortrait` или `FullScreen`

`statusbarOpacity` — прозрачность фона

`statusbarBackgroundColor` — цвет фона

`statusbarScheme` — цветовая схема для иконок

```
ApplicationWindow {  
    displayMode:  
        ApplicationDisplayMode.FullPortrait  
    statusBarForceVisible: true  
    statusBarOpacity: 1.0  
    statusBarBackgroundColor: "green"  
    statusBarScheme: Theme.LightOnDark  
}
```

Общие рекомендации

- Простая для понимания структура приложений
- Основная информация выделяется
 - позицией
 - цветом
 - размером
- Отзывчивый UI
- Различные макеты для различных экранов
- Автоматическая навигация для компонентов ввода
- Для прокрутки всегда нужен индикатор



Рекомендации для Silica

- Информативные, но не перегруженные обложки
- Статические элементы выделяются Theme.primaryColor
- Выбранные элементы выделяются Theme.highlightColor
- Поля слева и справа экрана Theme.horizontalPageMargin
- Нажимаемые элементы не меньше Theme.itemSizeMedium
- Не больше четырех элементов в вытягиваемых меню
- Если меню не доступно, его индикатора нет
- Диалоги не содержат кнопки "Accept" и "Reject"
- Значения свойств placeholderText и label определены
- Используются действия EnterKey



True Engineering

630128, г. Новосибирск,
ул. Кутателадзе, 4г

(383) 363-33-51, 363-33-50
info@trueengineering.ru
trueengineering.ru

**Новосибирский
Государственный
Университет**