Справочник Flutter



Составитель и переводчик

Александр Рэн

(Aleksandr Rhen)

Справочное руководство

по виджетам Flutter

2020 год

Оглавление

[О справочнике 3](#_Toc51926259)

[Основные понятия 3](#_Toc51926260)

[Каталог виджетов 7](#_Toc51926261)

[Виджеты основные 8](#_Toc51926262)

[Виджет Appbar 8](#_Toc51926263)

[Виджеты структуры и навигации 11](#_Toc51926264)

[Виджет Scaffold 11](#_Toc51926265)

[Виджет Floating action buttons (FAB) 17](#_Toc51926266)

# О справочнике

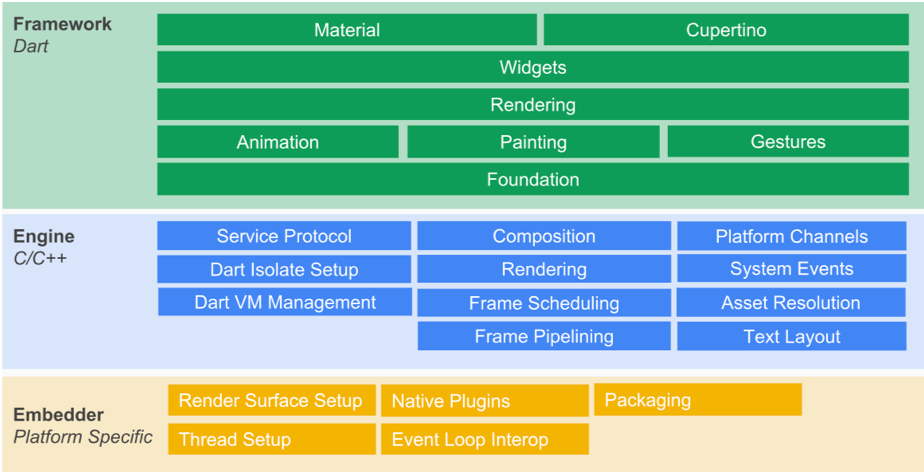
Данный справочник адресован помочь разработчику лучше ориентироваться в виджетах Flutter, благодаря чему код станет понятным, а *бесконечное* древо вложенности сможет обрести куда более лаконичный вид.

Издание разделено на несколько частей. **Первая часть** включает описание самого Flutter, его устройства и принципов работы. Понимание основ позволяет управлять логикой приложения. **Вторая часть** описывает наиболее распространённые виджеты Flutter, что позволит реализовать любой вариант интерфейса, а также его логику.

# Основные понятия

Если Flutter является для Вас дебютом в разработке программного обеспечения, то необходимо усвоить следующую терминологию:

* Flutter — SDK с открытым исходным кодом для создания мобильных приложений от компании Google. Он используется для разработки приложений под Android и iOS, а также это пока единственный способ разработки приложений под Google Fuchsia.
* Метод -



Когда мы пишем приложение Flutter, используя Dart, мы остаемся на уровне Flutter Framework (выделено зеленым цветом).

Flutter Framework взаимодействует с Flutter Engine (синим цветом) через слой абстракции, называемый Window. Этот уровень абстракции предоставляет ряд API для косвенного взаимодействия с устройством.

Также через этот уровень абстракции Flutter Engine уведомляет Flutter Framework, когда:

* событие, представляющее интерес, происходит на уровне устройства (изменение ориентации, изменение настроек, проблема с памятью, состояние работы приложения…)
* какое-то событие происходит на уровне стекла (=жест)
* канал платформы отправляет некоторые данные
* но также и в основном, когда Flutter Engine готов к рендерингу нового кадра

**Управление Flutter Framework рендерингом Flutter Engine**

В это сложно поверить, но это правда. За исключением некоторых случаев (cм. ниже) ни один код Flutter Framework не выполняется без запуска рендеринга Flutter Engine.

Исключения:

* Gesture / Жест (= событие на стекле)
* Сообщения платформы (= сообщения, которые создаются устройством, например, GPS)
* Сообщения устройства (= сообщения, которые относятся к изменению состояния устройства, например, ориентация, приложение, отправленное в фоновом режиме, предупреждения памяти, настройки устройства…)
* Future или http-ответы

(Между нами говоря, на самом деле можно применить визуальное изменение без вызова от Flutter Engine, но это не рекомендуется делать)

Вы меня спросите: "Если какой-то код, связанный с жестом, выполняется и вызывает визуальное изменение или если я использую timer для задания периодичности задачи, которая приводит к визуальным изменениям (например, анимация), то как это работает?"

Если вы хотите, чтобы произошло визуальное изменение или чтобы какой-то код выполнялся на основе таймера, то вам нужно сообщить Flutter Engine, что что-то должно быть отрисовано.

Обычно при следующем обновлении Flutter Engine обращается к Flutter Framework для выполнения некоторого кода и в конечном итоге предоставляет новую сцену для рендеринга.

**Краткое объяснение (более подробная информация будет позже):**

* Некоторые внешние события (жест, http-ответы и тд) или даже futures могут запускать задачи, которые приводят к необходимости обновления отображения. Соответствующее сообщение отправляется Flutter Engine (= Schedule Frame)
* Когда Flutter Engine готов приступить к обновлению рендеринга, он создает Begin Frame запрос
* Этот Begin Frame запрос перехватывается Flutter Framework, который выполняет задачи, связанные в основном с Tickers (например, анимацию)
* Эти задачи могут повторно создать запрос для более поздней отрисовки (пример: анимация не закончила своё выполнение, и для завершения ей потребуется получить еще один Begin Frame на более позднем этапе)
* Далее Flutter Engine отправляет Draw Frame, который перехватывается Flutter Framework, который будет искать любые задачи, связанные с обновлением макета с точки зрения структуры и размера
* После того, как все эти задачи выполнены, он переходит к задачам, связанным с обновлением макета с точки зрения отрисовки
* Если на экране есть что-то, что нужно нарисовать, то новая сцена (Scene) для визуализации отправляется в Flutter Engine, который обновит экран
* Затем Flutter Framework выполняет все задачи, которые будут выполняться после завершения рендеринга (= PostFrame callbacks), и любые другие последующие задачи, не связанные с рендерингом

… и этот процесс начинается снова и снова

Поэтому важный вопрос заключается в том, как движок Flutter организует все поведение приложения на основе рендеринга.

**RenderView и RenderObject**

Прежде чем погружаться в детали, связанные с потоком действий, самое время ввести понятие Rendering Tree.

Как уже говорилось ранее, всё в конечном итоге преобразуется в пиксели, которые будут отображаться на экране, и Flutter Framework преобразует Widgets, которые мы используем для разработки приложения, в визуальные блоки, которые будут отображаться на экране.

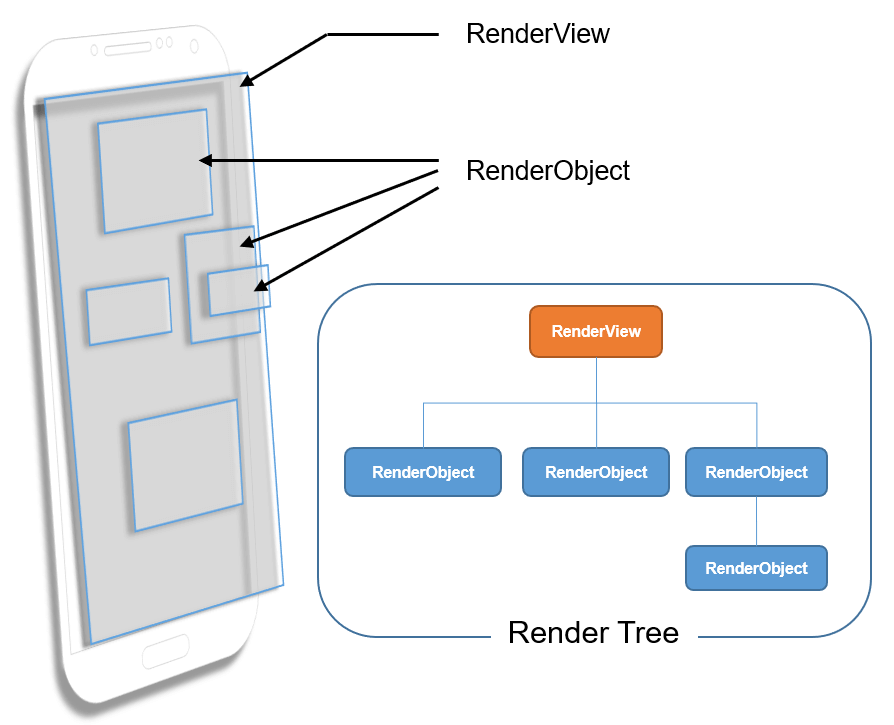
Данные визуальные части соответствуют объектам, называемым RenderObject, которые используются для:

определения некоторой области экрана с точки зрения размеров, положения, геометрии, а также с точки зрения «rendered content»

определения зон экрана, на которые могут повлиять жесты (= касания пальцев)

Набор всех RenderObject формирует дерево, называемое Render Tree. В верхней части этого дерева (= root) мы находим RenderView.

RenderView представляет общую поверхность для объектов Render Tree и является специальной версией RenderObject.



Cвязь между Widget и RenderObject будет рассмотрена далее. А пока пришло время немного углубиться…

Визуально мы могли бы представить все это следующим образом:

Чтобы вам получить представление о внутренних механизмах, посмотрите на следующую анимацию:

Для более эффективного понимания и применения строительных блоков Flutter, необходимо уяснить для себя ряд следующих основ:

* Основная точка входа

# Знакомство с виджетами

Виджеты Flutter построены с использованием современного framework’а, в основе которого лежит частичка идеологии React. Главный смысл заключается в том, что весь пользовательский интерфейс может быть представлен посредством разнообразных виджетов. Разработчик может управлять как внешним видом виджета, так и его состоянием. Если состояние виджета изменилось, виджет изменит своё представление (внешний вид).

Минимальное приложение на Flutter будет включать простой вызов runApp():

import 'package:flutter/material.dart';

void main() {

runApp(

Center(

child: Text(

'Hello, world!',

textDirection: TextDirection.ltr,

),

),

);

}

Приведённый пример показывает, что функция runApp() берёт виджет верхнего уровня и начинает отрисовывать древо остальных виджетов. В нашем примере древо виджетов содержит всего два элемента (виджета): Center и Text. Первый виджет (Center) скажет framework Flutter’у начать отрисовку древа виджетов по центру экрана, а второй виджет (Text) скажет, что текст должен содержать строчку текста «Привет, мир!» и иметь направление слева на право – об этом можно судить по свойству TextDirection: TextDirection.ltr, где “ltr[[1]](#footnote-1)”, очевидно угадывается, как «LeftToRight».

Важно! Все виджеты, которые мы используем при написании кода, будут относиться либо к классу StatelessWidget, либо к классу StatefulWidget. Виджет StatelessWidget предназначен

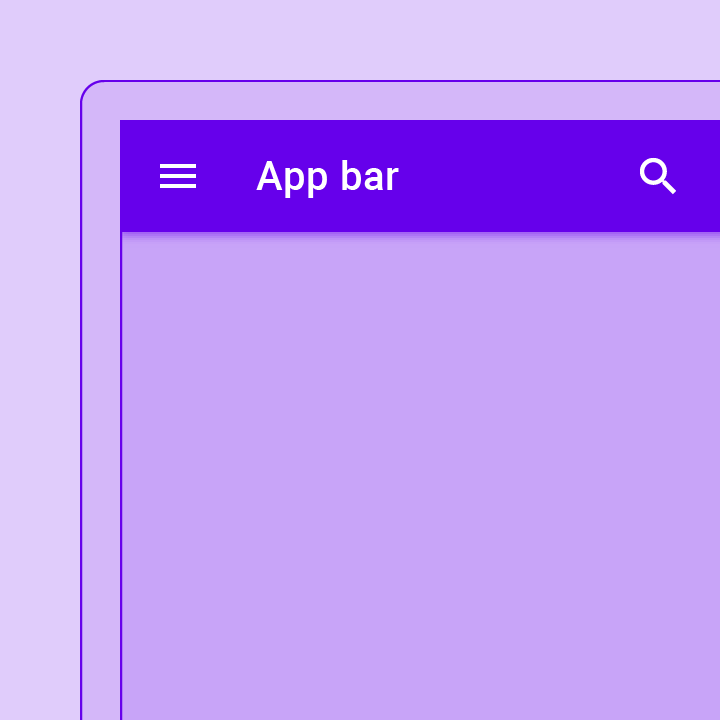
Затем framework Flutter рисует древо виджетов по центру экрана, в котором будет содержаться только одна строчка текста. Почему? Потому что мы

И здесь необходимо указать направление текста, для того, чтобы MaterialApp

# Каталог виджетов

## Виджеты основные

### Виджет Appbar

Данный виджет относится к «Material Design», состоит из панели инструментов («toolbar») и других возможных виджетов, таких как «TabBar» и «FlexibleSpaceBar».

Данный виджет зачастую используется как панель инструментов, и чаще всего мы можем его встретить в составе виджета «Scafford.appBar».

AppBar(

{Key key,

Widget leading,

bool automaticallyImplyLeading: true,

Widget title,

List<Widget> actions,

Widget flexibleSpace,

PreferredSizeWidget bottom,

double elevation,

Color shadowColor,

ShapeBorder shape,

Color backgroundColor,

Brightness brightness,

IconThemeData iconTheme,

IconThemeData actionsIconTheme,

TextTheme textTheme,

bool primary: true,

bool centerTitle,

bool excludeHeaderSemantics: false,

double titleSpacing: NavigationToolbar.kMiddleSpacing,

double toolbarOpacity: 1.0,

double bottomOpacity: 1.0,

double toolbarHeight}

)

проро

Виджеты доступности

Виджеты анимации и движений

Виджеты активности, графики и иконок

Виджеты асинхронной работы

Виджеты Cupertino

Виджеты ввода

Виджеты интерактивных моделей

Виджеты макетов / слоёв

## Виджеты структуры и навигации

### Виджет Scaffold



Краткое описание:

Scaffold реализует базовую структуру визуального макета.

Конструкторы:

Scaffold(

{Key key,

PreferredSizeWidget appBar,

Widget body,

Widget floatingActionButton,

FloatingActionButtonLocation floatingActionButtonLocation,

FloatingActionButtonAnimator floatingActionButtonAnimator,

List<Widget> persistentFooterButtons,

Widget drawer,

Widget endDrawer,

Widget bottomNavigationBar,

Widget bottomSheet,

Color backgroundColor,

bool resizeToAvoidBottomPadding,

bool resizeToAvoidBottomInset,

bool primary: true,

DragStartBehavior drawerDragStartBehavior:

DragStartBehavior.start,

bool extendBody: false,

bool extendBodyBehindAppBar: false,

Color drawerScrimColor,

double drawerEdgeDragWidth,

bool drawerEnableOpenDragGesture: true,

bool endDrawerEnableOpenDragGesture: true}).

Свойства:

[appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) → [PreferredSizeWidget](https://api.flutter.dev/flutter/widgets/PreferredSizeWidget-class.html)

An app bar to display at the top of the scaffold.

*final*

[backgroundColor](https://api.flutter.dev/flutter/material/Scaffold/backgroundColor.html) → [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)

The color of the [Material](https://api.flutter.dev/flutter/material/Material-class.html) widget that underlies the entire Scaffold. [[...]](https://api.flutter.dev/flutter/material/Scaffold/backgroundColor.html)

*final*

[body](https://api.flutter.dev/flutter/material/Scaffold/body.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

The primary content of the scaffold. [[...]](https://api.flutter.dev/flutter/material/Scaffold/body.html)

*final*

[bottomNavigationBar](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

A bottom navigation bar to display at the bottom of the scaffold. [[...]](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html)

*final*

[bottomSheet](https://api.flutter.dev/flutter/material/Scaffold/bottomSheet.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

The persistent bottom sheet to display. [[...]](https://api.flutter.dev/flutter/material/Scaffold/bottomSheet.html)

*final*

[drawer](https://api.flutter.dev/flutter/material/Scaffold/drawer.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

A panel displayed to the side of the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html), often hidden on mobile devices. Swipes in from either left-to-right ([TextDirection.ltr](https://api.flutter.dev/flutter/dart-ui/TextDirection-class.html)) or right-to-left ([TextDirection.rtl](https://api.flutter.dev/flutter/dart-ui/TextDirection-class.html)) [[...]](https://api.flutter.dev/flutter/material/Scaffold/drawer.html)

*final*

[drawerDragStartBehavior](https://api.flutter.dev/flutter/material/Scaffold/drawerDragStartBehavior.html) → [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior-class.html)

Determines the way that drag start behavior is handled. [[...]](https://api.flutter.dev/flutter/material/Scaffold/drawerDragStartBehavior.html)

*final*

[drawerEdgeDragWidth](https://api.flutter.dev/flutter/material/Scaffold/drawerEdgeDragWidth.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The width of the area within which a horizontal swipe will open the drawer. [[...]](https://api.flutter.dev/flutter/material/Scaffold/drawerEdgeDragWidth.html)

*final*

[drawerEnableOpenDragGesture](https://api.flutter.dev/flutter/material/Scaffold/drawerEnableOpenDragGesture.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Determines if the [Scaffold.drawer](https://api.flutter.dev/flutter/material/Scaffold/drawer.html) can be opened with a drag gesture. [[...]](https://api.flutter.dev/flutter/material/Scaffold/drawerEnableOpenDragGesture.html)

*final*

[drawerScrimColor](https://api.flutter.dev/flutter/material/Scaffold/drawerScrimColor.html) → [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)

The color to use for the scrim that obscures primary content while a drawer is open. [[...]](https://api.flutter.dev/flutter/material/Scaffold/drawerScrimColor.html)

*final*

[endDrawer](https://api.flutter.dev/flutter/material/Scaffold/endDrawer.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

A panel displayed to the side of the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html), often hidden on mobile devices. Swipes in from right-to-left ([TextDirection.ltr](https://api.flutter.dev/flutter/dart-ui/TextDirection-class.html)) or left-to-right ([TextDirection.rtl](https://api.flutter.dev/flutter/dart-ui/TextDirection-class.html)) [[...]](https://api.flutter.dev/flutter/material/Scaffold/endDrawer.html)

*final*

[endDrawerEnableOpenDragGesture](https://api.flutter.dev/flutter/material/Scaffold/endDrawerEnableOpenDragGesture.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Determines if the [Scaffold.endDrawer](https://api.flutter.dev/flutter/material/Scaffold/endDrawer.html) can be opened with a drag gesture. [[...]](https://api.flutter.dev/flutter/material/Scaffold/endDrawerEnableOpenDragGesture.html)

*final*

[extendBody](https://api.flutter.dev/flutter/material/Scaffold/extendBody.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

If true, and [bottomNavigationBar](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html) or [persistentFooterButtons](https://api.flutter.dev/flutter/material/Scaffold/persistentFooterButtons.html) is specified, then the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html) extends to the bottom of the Scaffold, instead of only extending to the top of the [bottomNavigationBar](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html) or the [persistentFooterButtons](https://api.flutter.dev/flutter/material/Scaffold/persistentFooterButtons.html). [[...]](https://api.flutter.dev/flutter/material/Scaffold/extendBody.html)

*final*

[extendBodyBehindAppBar](https://api.flutter.dev/flutter/material/Scaffold/extendBodyBehindAppBar.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

If true, and an [appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) is specified, then the height of the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html) is extended to include the height of the app bar and the top of the body is aligned with the top of the app bar. [[...]](https://api.flutter.dev/flutter/material/Scaffold/extendBodyBehindAppBar.html)

*final*

[floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

A button displayed floating above [body](https://api.flutter.dev/flutter/material/Scaffold/body.html), in the bottom right corner. [[...]](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html)

*final*

[floatingActionButtonAnimator](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButtonAnimator.html) → [FloatingActionButtonAnimator](https://api.flutter.dev/flutter/material/FloatingActionButtonAnimator-class.html)

Animator to move the [floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) to a new [floatingActionButtonLocation](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButtonLocation.html). [[...]](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButtonAnimator.html)

*final*

[floatingActionButtonLocation](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButtonLocation.html) → [FloatingActionButtonLocation](https://api.flutter.dev/flutter/material/FloatingActionButtonLocation-class.html)

Responsible for determining where the [floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) should go. [[...]](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButtonLocation.html)

*final*

[*hashCode*](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object. [[...]](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html)

*@nonVirtual, read-only, inherited*

[*key*](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)

Controls how one widget replaces another widget in the tree. [[...]](https://api.flutter.dev/flutter/widgets/Widget/key.html)

*final, inherited*

[persistentFooterButtons](https://api.flutter.dev/flutter/material/Scaffold/persistentFooterButtons.html) → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)>

A set of buttons that are displayed at the bottom of the scaffold. [[...]](https://api.flutter.dev/flutter/material/Scaffold/persistentFooterButtons.html)

*final*

[primary](https://api.flutter.dev/flutter/material/Scaffold/primary.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether this scaffold is being displayed at the top of the screen. [[...]](https://api.flutter.dev/flutter/material/Scaffold/primary.html)

*final*

[resizeToAvoidBottomInset](https://api.flutter.dev/flutter/material/Scaffold/resizeToAvoidBottomInset.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

If true the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html) and the scaffold's floating widgets should size themselves to avoid the onscreen keyboard whose height is defined by the ambient [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html)'s [MediaQueryData.viewInsets](https://api.flutter.dev/flutter/widgets/MediaQueryData/viewInsets.html) bottom property. [[...]](https://api.flutter.dev/flutter/material/Scaffold/resizeToAvoidBottomInset.html)

*final*

[~~resizeToAvoidBottomPadding~~](https://api.flutter.dev/flutter/material/Scaffold/resizeToAvoidBottomPadding.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

This flag is deprecated, please use [resizeToAvoidBottomInset](https://api.flutter.dev/flutter/material/Scaffold/resizeToAvoidBottomInset.html) instead. [[...]](https://api.flutter.dev/flutter/material/Scaffold/resizeToAvoidBottomPadding.html)

*@*[*Deprecated*](https://api.flutter.dev/flutter/dart-core/Deprecated-class.html)*('Use resizeToAvoidBottomInset to specify if the body should resize when the keyboard appears. ' 'This feature was deprecated after v1.1.9.'), final*

[*runtimeType*](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

*read-only, inherited*

Методы:

[*createElement*](https://api.flutter.dev/flutter/widgets/StatefulWidget/createElement.html)() → [StatefulElement](https://api.flutter.dev/flutter/widgets/StatefulElement-class.html)

Creates a [StatefulElement](https://api.flutter.dev/flutter/widgets/StatefulElement-class.html) to manage this widget's location in the tree. [[...]](https://api.flutter.dev/flutter/widgets/StatefulWidget/createElement.html)

*inherited*

[createState](https://api.flutter.dev/flutter/material/Scaffold/createState.html)() → [ScaffoldState](https://api.flutter.dev/flutter/material/ScaffoldState-class.html)

Creates the mutable state for this widget at a given location in the tree. [[...]](https://api.flutter.dev/flutter/material/Scaffold/createState.html)

*override*

[*debugDescribeChildren*](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/debugDescribeChildren.html)() → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)>

Returns a list of [DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html) objects describing this node's children. [[...]](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/debugDescribeChildren.html)

*@protected, inherited*

[*debugFillProperties*](https://api.flutter.dev/flutter/widgets/Widget/debugFillProperties.html)([DiagnosticPropertiesBuilder](https://api.flutter.dev/flutter/foundation/DiagnosticPropertiesBuilder-class.html) properties) → void

Add additional properties associated with the node. [[...]](https://api.flutter.dev/flutter/widgets/Widget/debugFillProperties.html)

*inherited*

[*noSuchMethod*](https://api.flutter.dev/flutter/dart-core/Object/noSuchMethod.html)([Invocation](https://api.flutter.dev/flutter/dart-core/Invocation-class.html) invocation) → dynamic

Invoked when a non-existent method or property is accessed. [[...]](https://api.flutter.dev/flutter/dart-core/Object/noSuchMethod.html)

*inherited*

[*toDiagnosticsNode*](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) name, [DiagnosticsTreeStyle](https://api.flutter.dev/flutter/foundation/DiagnosticsTreeStyle-class.html) style}) → [DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)

Returns a debug representation of the object that is used by debugging tools and by [DiagnosticsNode.toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticsNode/toStringDeep.html). [[...]](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)

*inherited*

[*toString*](https://api.flutter.dev/flutter/foundation/Diagnosticable/toString.html)({[DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel-class.html) minLevel: DiagnosticLevel.info}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a string representation of this object.

*inherited*

[*toStringDeep*](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringDeep.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) prefixLineOne: '', [String](https://api.flutter.dev/flutter/dart-core/String-class.html) prefixOtherLines, [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel-class.html) minLevel: DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a string representation of this node and its descendants. [[...]](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringDeep.html)

*inherited*

[*toStringShallow*](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringShallow.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) joiner: ', ', [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel-class.html) minLevel: DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a one-line detailed description of the object. [[...]](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringShallow.html)

*inherited*

[*toStringShort*](https://api.flutter.dev/flutter/widgets/Widget/toStringShort.html)() → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A short, textual description of this widget.

*inherited*

Операторы:

[*operator ==*](https://api.flutter.dev/flutter/widgets/Widget/operator_equals.html)([Object](https://api.flutter.dev/flutter/dart-core/Object-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

The equality operator. [[...]](https://api.flutter.dev/flutter/widgets/Widget/operator_equals.html)

*@nonVirtual, inherited*

Статические методы:

[geometryOf](https://api.flutter.dev/flutter/material/Scaffold/geometryOf.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context) → [ValueListenable](https://api.flutter.dev/flutter/foundation/ValueListenable-class.html)<[ScaffoldGeometry](https://api.flutter.dev/flutter/material/ScaffoldGeometry-class.html)>

Returns a [ValueListenable](https://api.flutter.dev/flutter/foundation/ValueListenable-class.html) for the [ScaffoldGeometry](https://api.flutter.dev/flutter/material/ScaffoldGeometry-class.html) for the closest [Scaffold](https://api.flutter.dev/flutter/material/Scaffold-class.html) ancestor of the given context. [[...]](https://api.flutter.dev/flutter/material/Scaffold/geometryOf.html)

[hasDrawer](https://api.flutter.dev/flutter/material/Scaffold/hasDrawer.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) registerForUpdates: true}) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

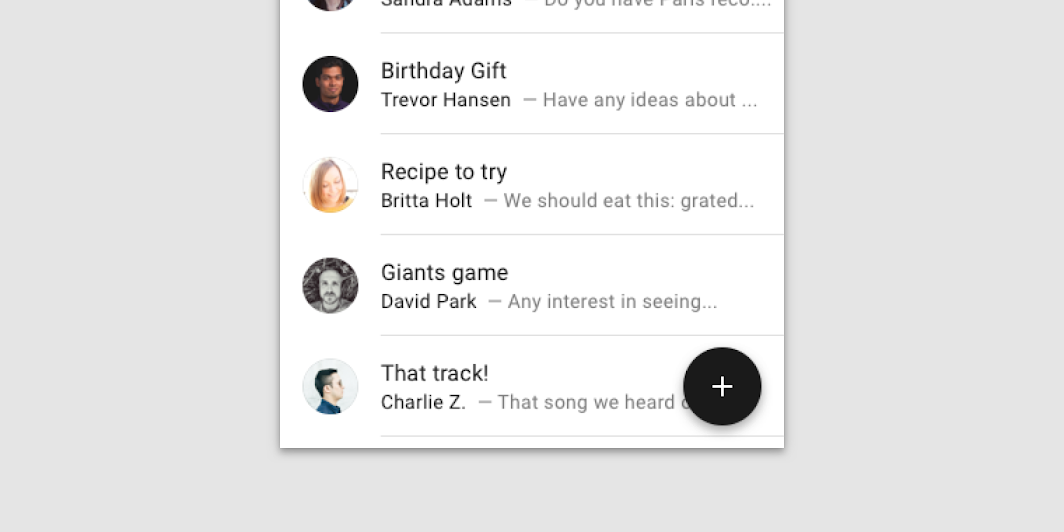
Whether the Scaffold that most tightly encloses the given context has a drawer. [[...]](https://api.flutter.dev/flutter/material/Scaffold/hasDrawer.html)

[of](https://api.flutter.dev/flutter/material/Scaffold/of.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) nullOk: false}) → [ScaffoldState](https://api.flutter.dev/flutter/material/ScaffoldState-class.html)

The state from the closest instance of this class that encloses the given context. [[...]](https://api.flutter.dev/flutter/material/Scaffold/of.html)

### Виджет Floating action buttons (FAB)

Плавающая кнопка действия.



Краткое описание:

Floating action buttons (FAB) предназначена для вызова основных или наиболее распространённых действий на экране. Данный виджет отображается поверх остального содержимого на экране и имеет по умолчанию круглую кнопку со значком в центре «+».

Для использования данного виджета необходимо убедиться, что импортирован следующий макет:

package:flutter/material.dart.

Специальные возможности

API Flutter поддерживает настройку специальных возможностей, включающих поддержку крупного шрифта, программное обеспечение для чтения с экрана и высокую контрастность. Для получения дополнительной информации необходимо перейти по ссылке:

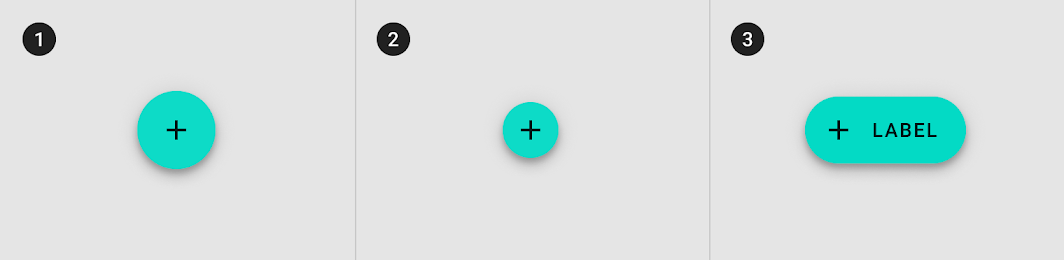
https://flutter.dev/docs/development/accessibility-and-localization/accessibility

https://flutter.dev/docs/development/accessibility-and-localization/internationalization

Вариации представления FAB

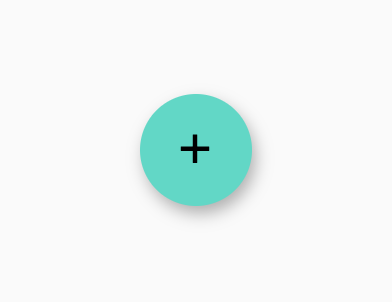
Существуют три типа FAB:

1. Обычная кнопка
2. Минимизированная кнопка
3. Расширенная кнопка



Обычная FAB

Обычные FAB – это FAB, имеющие стандартный размер и не включающие текст.



Образец кода:

FloatingActionButton(

backgroundColor: const Color(0xff03dac6),

foregroundColor: Colors.black,

onPressed: () {

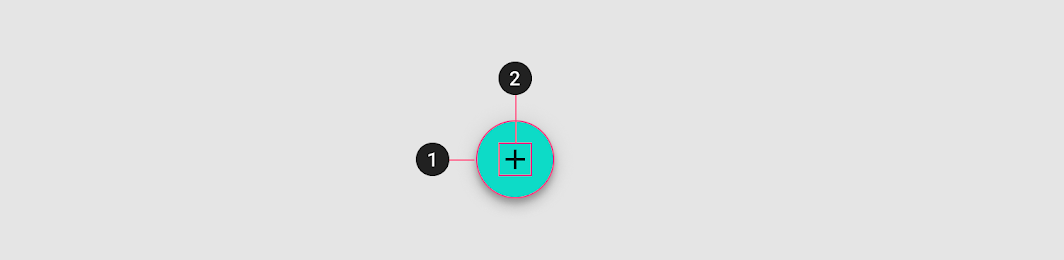
// Результат отклика при нажатии на кнопку

},

child: Icon(Icons.add),

)

Структура и ключевые свойства для обычной FAB



Обычная FAB представляет собой (1) контейнер и (2) иконку.

Свойства контейнера

| Свойство |
| --- |
| **Color** | backgroundColor |
| **Stroke color** | color of side on a custom shape |
| **Stroke width** | width of side on a custom shape |
| **Shape** | shape |
| **Elevation** | elevation |
| **Ripple color** | splashColor |

Свойства иконки

| Свойство |
| --- |
| **Icon** | child |
| **Color** | foregroundColor |
| **Size** | size on child when using an Icon |

Виджеты рисования и эффектов

Виджеты пролистывания

Виджеты стиля

Виджеты для текста

1. Допускается и обратный вариант, когда текст представлен справа-налево. TextDirection: TextDirection.rtl. Более подробнее об этом указано здесь: https://api.flutter.dev/flutter/dart-ui/TextDirection-class.html [↑](#footnote-ref-1)