Flutter

Widget:

The central idea is that you build your UI out of widgets. Widgets describe what their view should look like given their current configuration and state. When a widget's state changes, the widget rebuilds its description, which the framework differs against the previous description in order to determine the minimal changes needed in the underlying render tree to transition from one state to the next. When writing an app, you'll commonly author new widgets that are subclasses of either StatelessWidget or StatefulWidget, depending on whether your widget manages any state. A widget's main job is to implement a build() function, which describes the widget in terms of other, lower-level widgets.

Basic widgets

Flutter comes with a suite of powerful basic widgets, of which the following are commonly used:

Text

The Text widget lets you create a run of styled text within your application.

Row, Column

These flex widgets let you create flexible layouts in both the horizontal (Row) and vertical (Column) directions. The design of these objects is based on the web's flexbox layout model.

Stack

Instead of being linearly oriented (either horizontally or vertically), a Stack widget lets you place widgets on top of each other in paint order. You can then use the Positioned widget on children of a Stack to position them relative to the top, right, bottom, or left edge of the stack. Stacks are based on the web's absolute positioning layout model.

Container

The Container widget lets you create a rectangular visual element. A container can be decorated with a BoxDecoration, such as a background, a border, or a shadow. A Container can also have margins, padding, and constraints applied to its size. In addition, a Container can be transformed in three dimensional space using a matrix.

Material App

Many Material Design widgets need to be inside of a MaterialApp to display properly, in order to inherit theme data. Therefore, run the application with a MaterialApp. MaterialApp is a widget that introduces many interesting tools such as Navigator or Theme to help you develop your app.The MaterialApp widget provides a ton of benefits that affect its entire widget subtree. This section is the beginning of our discussion of many widgets that provide helpful functionality for free.

MaterialApp is even more convenient than WidgetsApp. It adds Material Design-specific functionality and styling options to your app. It doesn't just *help* set up the Navigator, it does it for you. If you use the Material app widget, you don't have to worry about implementing the animations that happen when a user navigates between pages: the widget takes care of that for you. It also allows you to use widgets that are specifically in the Material widgets collection, and there are plenty of those.

It's called a *Material app* because it leans on Material style guidelines For example, page animations from one route to another are designed as you'd expect on an Android device. And all of the widgets in the Material widget library have that standard Google look and feel. The MaterialApp widget provides quite a bit of convenience, but everything is reversible.

Scaffold

A Scaffold Widget provides a framework which implements the basic material design visual layout structure of the flutter app. It provides APIs for showing drawers, snack bars and bottom sheets. Scaffold contains various functionality from giving an appbar, a floating button, a drawer, background color, bottom navigation bar, footer buttons,body.

- appBar

An *appBar* is to display at the top of the scaffold it's one of the primary contents of Scaffold without which a scaffold widget is incomplete. It defines what has to be displayed at the top of the screen. appBar uses the AppBar widget properties through Scaffold.appBar. This AppBar widget itself has various properties like title, elevation,brightness etc to name a few.

- body

body is the other primary and minimum required property of the scaffold which signifies the area below the app bar and behind the floatingActionButton and drawer. Any widget in the *body of the scaffold* is positioned at the top-left corner by default.

- floatingActionButton

A floatingActionButton is a button displayed floating above the body in the bottom right corner. It is a circular icon button that hovers over content to promote a primary action in the application. floatingActionButton uses the FloatingActionButton widget properties through Scaffold.floatingActionButton.

- drawer

A *drawer* is a panel displayed to the side of the body, often hidden on mobile devices. One usually has to swipe left to right or right to left to access the drawer.
It uses the Drawer widget properties which is a material design panel that slides in horizontally from the edge of a Scaffold to show navigation links in an application and this widget is used in scaffold using Scaffold.drawer property.

- persistentFooterButtons

persistentFooterButtons is a list of buttons that are displayed at the bottom of the scaffold. These buttons are persistently visible, even if the body of the scaffold scrolls. They will be wrapped in a ButtonBar and are rendered above the bottomNavigationBar but below the body.

- bottomNavigationBar

bottomNavigationBar is used to display a navigation bar at the bottom of the scaffold. It is rendered below the persistentFooterButtons and the body.

endDrawer

An endDrawer is like a drawer property but in endDrawer by default the drawer is displayed at the right side of the screen.

primary

Whether the scaffold is being displayed at the top of the screen. If true then the height of the appBar will be extended by the height of the screen's status bar, i.e. the top padding for. The default value of this property is true.

backgroundColor

This property sets the background color of the scaffold.

backgroundColor: Colors.white,

resizeToAvoidBottomInset

If this property is true the body and the scaffold's floating widgets should size themselves to avoid the onscreen keyboard whose height is defined by the bottom property.

For example, if there is an onscreen keyboard displayed above the scaffold, the body can be resized to avoid overlapping the keyboard, which prevents widgets inside the body from being obscured by the keyboard. Defaults to true.