The bases and basics of a flutter project

AP-FALL(1403-1404)-DR. VAHIDI

BY NAZANIN FOROUTAN

What is flutter and why do we use it?

QUICK REVIEW

Flutter is an open source framework developed and supported by Google. Frontend and full-stack developers use Flutter to build an application's user interface (UI) for multiple platforms with a single codebase.

Flutter is often used with DART, which is an object-oriented programming language.

Why flutter?

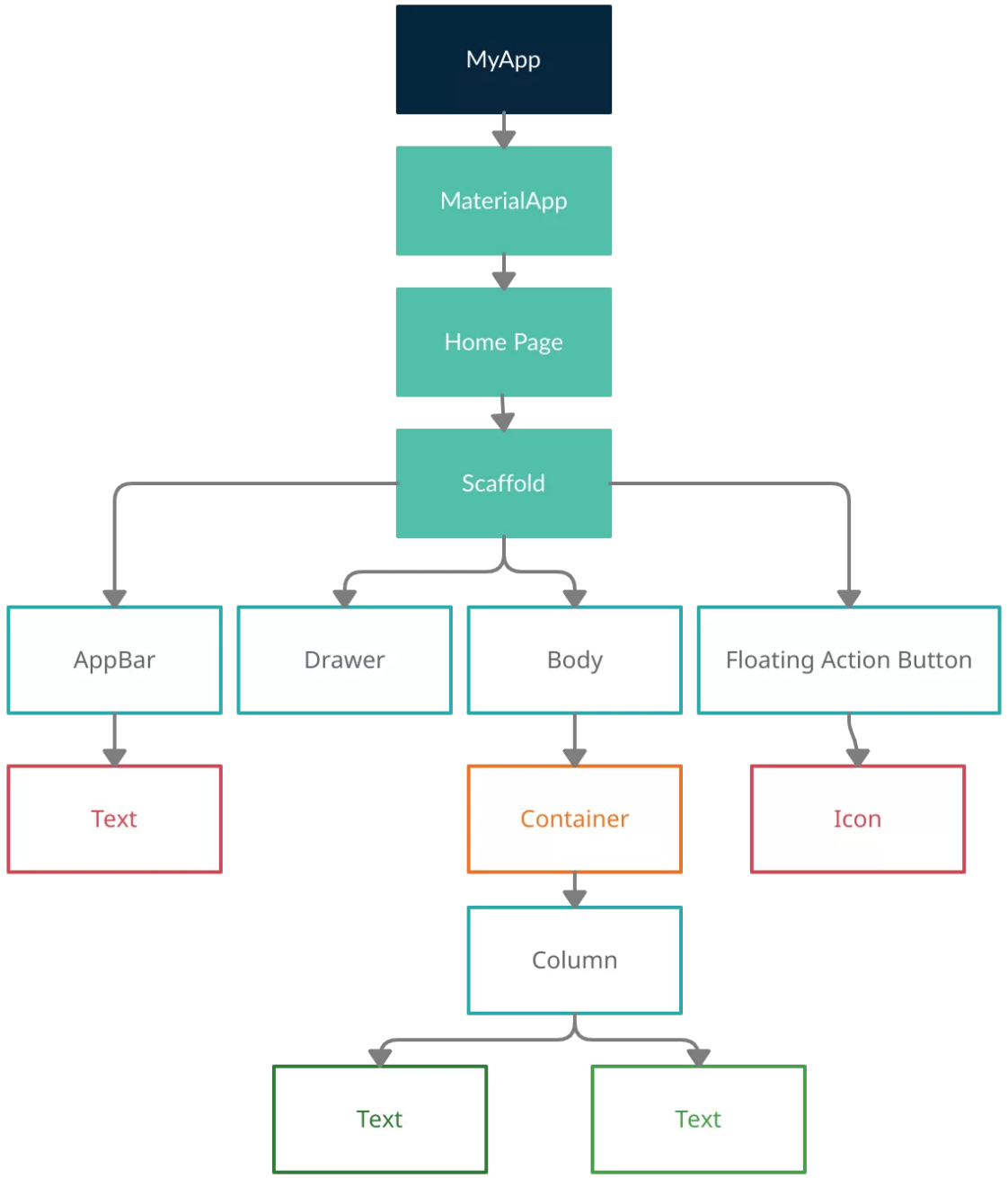
- Open source
- Cross-Platform Consistency
- Strong community
- Fast Development with Hot Reload
- High Performance

What are widgets?

A widget is a key component of the user interface (UI) in Flutter. Widgets are used to describe the structure and layout of your application's UI elements, such as buttons, text fields, images, containers, and more.

Widgets are classes and everything in flutter is a widget.

Widget tree

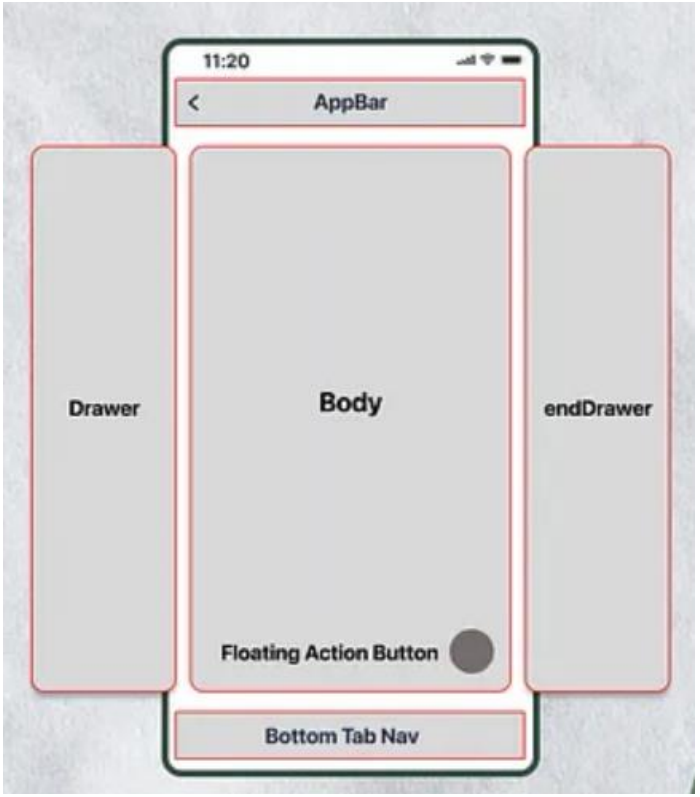

MyApp
MaterialApp
Home Page
Scaffold
AppBar
Drawer
Body
Floating Action Button
Text
Container
Icon
Column
Text
Text

Material design?

- Material is an adaptable design system, backed by open-source code, that helps developers easily build high-quality, digital experiences. Material design is the default design language in flutter and feel across Android devices and web applications.
- Cupertino, on the other hand, is Apple's design system.
- To access Material Design components, we should use the MaterialApp class, which provides access to Material Design widgets and themes.

Scaffold

- Scaffold is a class in flutter which provides many widgets or we can say APIs like Drawer, Snack-Bar, Bottom-Navigation-Bar, Floating-Action-Button, App-Bar, etc. Scaffold will expand or occupy the whole device screen. It will occupy the available space.



Stateless vs Stateful

- Stateless Widget is a type of widget which once built , then it’s properties and state can’t be changed. These widgets are immutable, once created can’t be modified.
- Stateful Widgets is a type of widget that can change state. It can maintain and update the appearance in the response to change in state.

Stateless and Stateful syntax

- **StateLess**

```
class myStateLessWidget extends StatelessWidget{
     const myStateLessWidget({super.key});

     @override
     Widget build(BuildContext context) {
       return const Text('data');
     }
   }
```

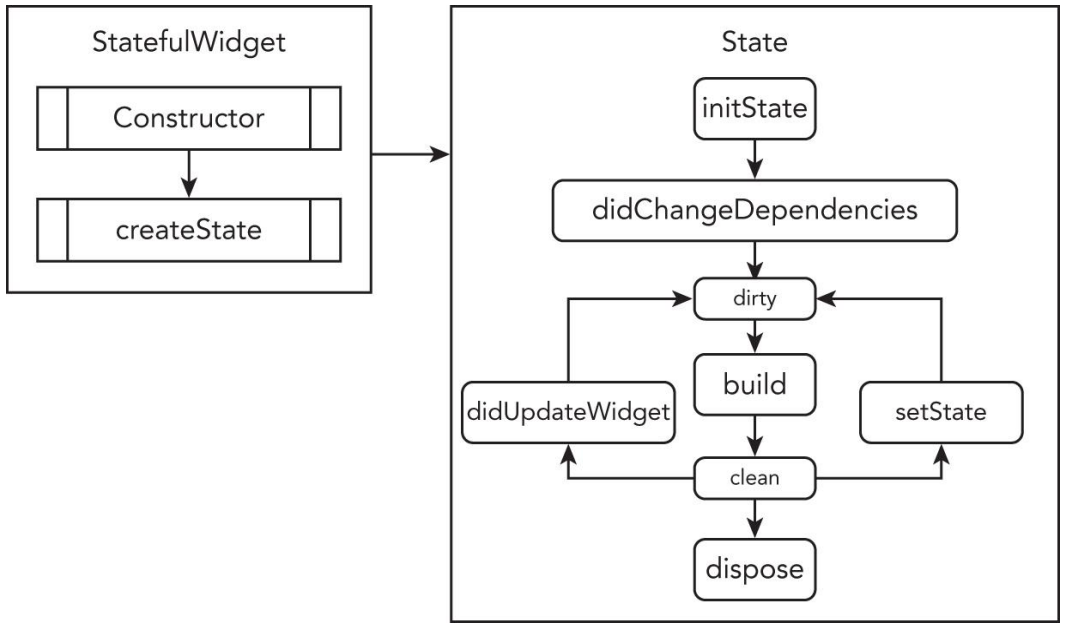
- **Stateful**

```
class myStatefulWidget extends StatefulWidget {
     const myStatefulWidget({super.key});

     @override
     State<myStatefulWidget> createState() =>
   _myStatefulWidgetState();
   }

   class _myStatefulWidgetState extends State<myStatefulWidget>
   {
     @override
     Widget build(BuildContext context) {
       return Text('data');
     }
   }
```

Widgets lifecycle



- **Stateless**

Stateless widgets do not have a lifecycle beyond their creation. They are created, displayed, and then destroyed when no longer needed.

- **Stateful**

Stateful widgets, on the other hand, are more complex because they maintain state over time and can update dynamically. The lifecycle of a stateful widget consists of several stages, and it revolves around the **State** object that's associated with the widget.

Introduction to common flutter widgets

1. **Basic Widgets**

These are the most fundamental widgets in Flutter that form the building blocks of a UI.

- **Text**: Displays a string of text.
- **Row**: Lays out its children in a horizontal array.
- **Column**: Lays out its children in a vertical array.
- **Container**: A versatile widget for adding padding, margins, and background color to its child.
- **Center**: Centers its child within itself.
- **Icon**: Displays an icon from the Material Icons library.
- **Image**: Displays an image from various sources (network, file, asset).
- **Placeholder**: A box that represents where other widgets will eventually appear.

2. **Input Widgets**

Widgets that accept user inputs.

- **TextField**: A text input widget.
- **Checkbox**: A box that can be checked or unchecked.
- **Radio**: Allows the user to select one option from a set.
- **Switch**: A toggle button to select between on and off states.
- **Slider**: A slider for selecting a value from a range of values.
- **DropdownButton**: A button that shows a dropdown menu of items.
- **InkWell**: A rectangular area that responds to touch, showing a ripple effect.
- **GestureDetector**: Detects gestures like taps, drags, and swipes.

3. **Layout Widgets**

These widgets are used to arrange and align other widgets.

- **Padding**: Adds padding to its child.
- **Align**: Aligns its child within itself.
- **Expanded**: Expands a child of a Row, Column, or Flex to fill available space.
- **Flexible**: Similar to Expanded, but allows flexibility in the size.
- **Stack**: Stacks its children on top of each other.
- **Wrap**: Wraps its children in multiple horizontal or vertical runs.
- **ListView**: A scrollable list of widgets arranged linearly.
- **GridView**: A scrollable, 2D array of widgets.

4. **Navigation Widgets**

Widgets related to app navigation and routing.

- **Navigator**: A widget that manages a stack of route pages.
- **Scaffold**: Implements the basic visual structure of a Material Design screen.
- **AppBar**: A material design app bar.
- **BottomNavigationBar**: Displays a bar at the bottom for navigating between different screens.
- **TabBar**: Displays tabs horizontally.
- **Drawer**: A material design panel that slides in horizontally from the edge of the screen.

5. **Styling & Aesthetic Widgets**

These widgets are related to styling and enhancing the look and feel of your app.

- **DecoratedBox**: A widget that paints a decoration behind its child.
- **Opacity**: Changes the opacity of a child widget.
- **FittedBox**: Scales and positions its child within itself.
- **ClipRRect**: Clips its child using a rounded rectangle.
- **Theme**: Inherits the app's theme and allows setting the theme for child widgets.
- **Divider**: Draws a line (horizontal or vertical) to divide widgets.

6. **Scrolling Widgets**

These widgets enable scrolling behavior.

- **SingleChildScrollView**: A box that scrolls a single child.
- **ListView**: Scrollable list of widgets arranged linearly.
- **GridView**: A scrollable grid of widgets.
- **ScrollView**: A base class for scrollable views.
- **CustomScrollView**: A scrollable view with a custom scroll behavior.

7. **Animation & Transition Widgets**

Widgets for adding animations and transitions.

- **AnimatedContainer**: A container that animates changes to its properties.
- **AnimatedOpacity**: Animates changes to the opacity of a widget.
- **AnimatedCrossFade**: Crossfades between two children.
- **Hero**: Implements a hero animation for page transitions.
- **FadeTransition**: Fades in or out a widget over a duration.
- **RotationTransition**: Rotates a widget by a specific angle.
- **ScaleTransition**: Scales a widget.
- **AnimatedBuilder**: A widget that allows creating custom animations.

8. **Dialogs & Popups**

Widgets for displaying popups and dialogs.

- **AlertDialog**: A material design alert dialog.
- **SimpleDialog**: A basic dialog for showing a list of options.
- **SnackBar**: A lightweight message bar that displays at the bottom of the screen.
- **BottomSheet**: A panel that slides up from the bottom of the screen.

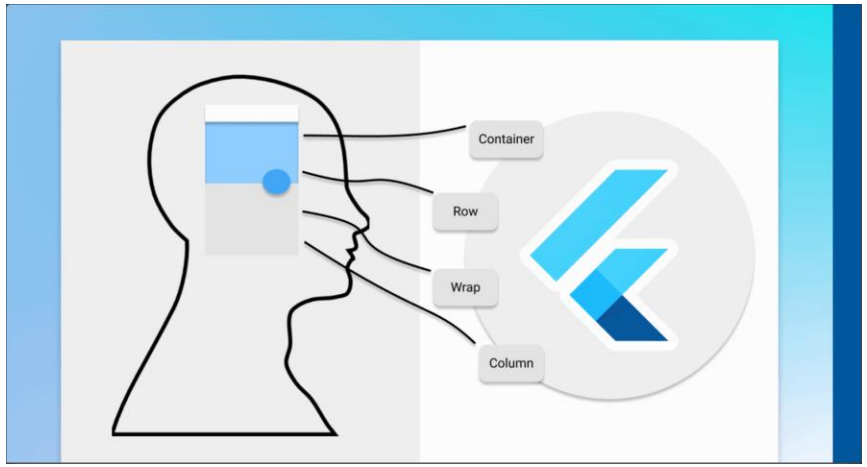
How to work with widgets?

LET'S BREAK DOWN 3 WIDGETS TO FIND IT OUT

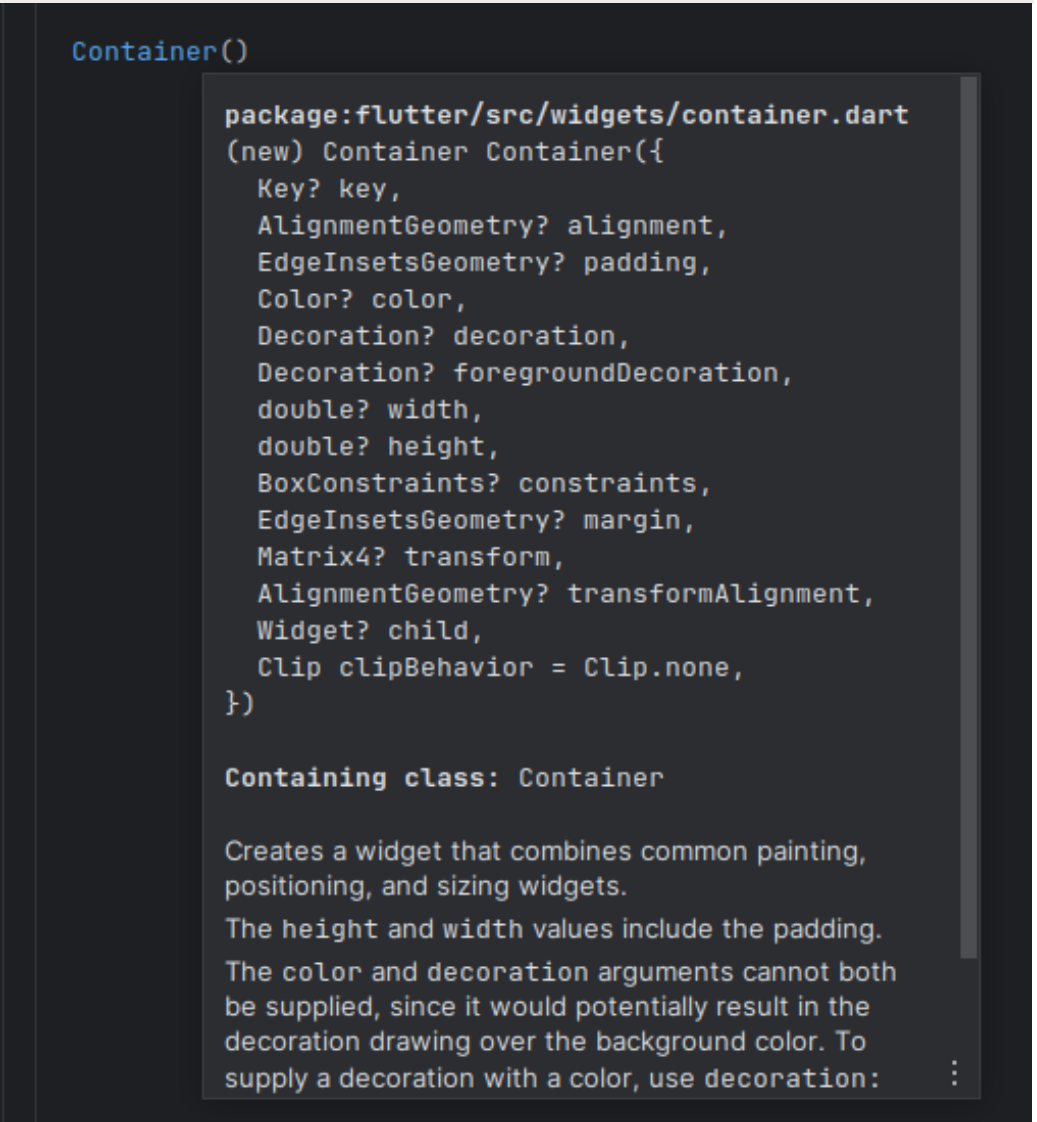
- **Ctrl + space** for more properties
- **Hold on!**
- **Ctrl + click** on the name of class for get more info about what are dealing with exactly?

General info about widgets

- As mentioned earlier, widgets are classes, and everything you see in Flutter is a widget. Every single page in Flutter is a combination of widgets, organized into a widget tree.
- Each widget comes with a set of fields (properties) that define its behavior, appearance, and structure.



1. Container widget



Container is a widget class that allows you to customize its child widget. Some of the most useful properties of a container widgets are:

- **Child**: The child field in a Container widget is used to define the widget that will be placed inside the container.
- **Color**: Specifies the background color of the container
- **Decoration**: It allows you to customize the appearance of the container, such as its background, borders, shadows, gradients, and shapes.
- **Padding and margin**: Padding the area inside the widget and the child, margin the area outside (the one surrounds the container)
- **Width and height**: Clear I guess:))

2. AppBar widget

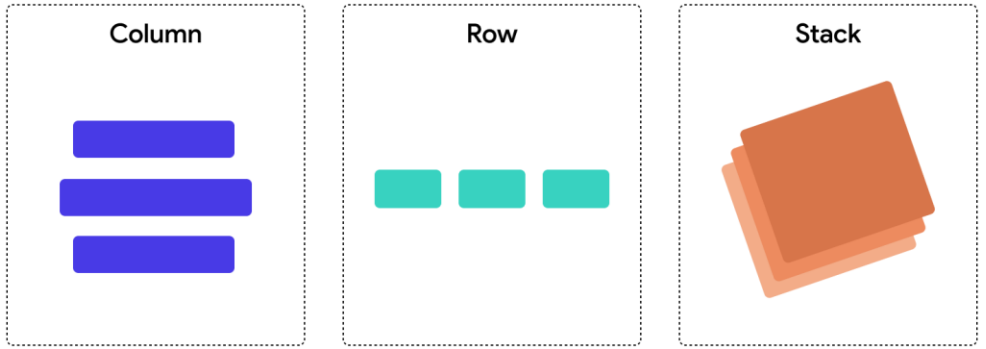
```
AppBar()
    Key? key,
    Widget? leading,
    bool automaticallyImplyLeading = tru
    Widget? title,
    List<Widget>? actions,
    Widget? flexibleSpace,
    PreferredSizeWidget? bottom,
    double? elevation,
    double? scrolledUnderElevation,
```

The top app bar provides content and actions related to the current screen. It can transform into a contextual action bar. Some of the most useful properties of a container widgets are:

- **Leading**: It is used to specify a widget that appears at the start of the app bar.
- **Actions**: It is used to define a list of widgets on the right side of the app bar. They usually represent actions the user can take, like search, settings, or other app-specific actions.

3. Column, Row and stack

- **Column**: It is a layout widget that arranges its children in a **vertical** direction.
- **Row**: It is used to arranges its children **horizontally.**
- **Stack:** allows you to position its children **on top of each other**, essentially stacking them in layers.



Useful Links

- Flutter documentation
- GeeksforGeeks
- Net-Ninja YouTube playlist (flutter tutorial)
- An example of login/sign-up form
- Flutter widgets playlist
- Flutter clean architecture playlist

sources

- What is Flutter?
- Widgets in Flutter: A Brief Overview | by Techdynasty - Medium
- Mastering Flutter's Scaffold Widget: A Comprehensive ...
- What is Widgets in Flutter?
- Rows, Column & Stack | FlutterFlow Documentation
- Widgets in Flutter: A Brief Overview | by Techdynasty - Medium
- Flutter - Material Design
- All About Flutter