

Flutter & Dart Complete Guide

Prepared by Eng. Yousef Salah Kassem

DAFL 2
Flutter UI Part 1

Lecture 1



Chapter

01

Table Of Content

- Introduction
- Flutter Architecture
- Widgets
- Material



Chapter
01**Introduction**

The Flutter architecture mainly comprises of four components.

- Flutter Engine
- Foundation Library
- Widgets
- Design Specific Widgets



Chapter
01**Flutter Engine**

- It is a portable runtime for high-quality mobile apps and primarily based on the C++ language.
- It implements Flutter core libraries that include animation and graphics, file and network I/O, plugin architecture, accessibility support, and a dart runtime for developing, compiling, and running Flutter applications.



Chapter
01**Foundation Library**

- It contains all the required packages for the basic building blocks of writing a Flutter application.
- These libraries are written in Dart language.



Chapter
01

Widgets

- In Flutter, everything is a widget, which is the core concept of this framework. Widget in the
- The widgets are similar to the React components.
- Flutter is basically a user interface component that affects and controls the view and interface of the app.
- It represents an immutable description of part of the user interface and includes graphics, text, shapes, and animations that are created using widgets.



Chapter
01**Widgets**

- In Flutter, the application is itself a widget that contains many sub widgets.
- It means the app is the top-level widget, and its UI is build using one or more children widgets, which again includes sub child widgets.
- This feature helps you to create a complex user interface very easily.



Chapter
01**Design Specific Widgets**

- The Flutter framework has two sets of widgets that conform to specific design languages.
- These are Material Design for Android application and Cupertino Style for IOS application.



Everything is a Widget !



Widgets

Sized Box

- ◆ Empty Box to Add elements or objects inside of it.
- ◆ Used as a spacer between objects or elements with fixed sizes

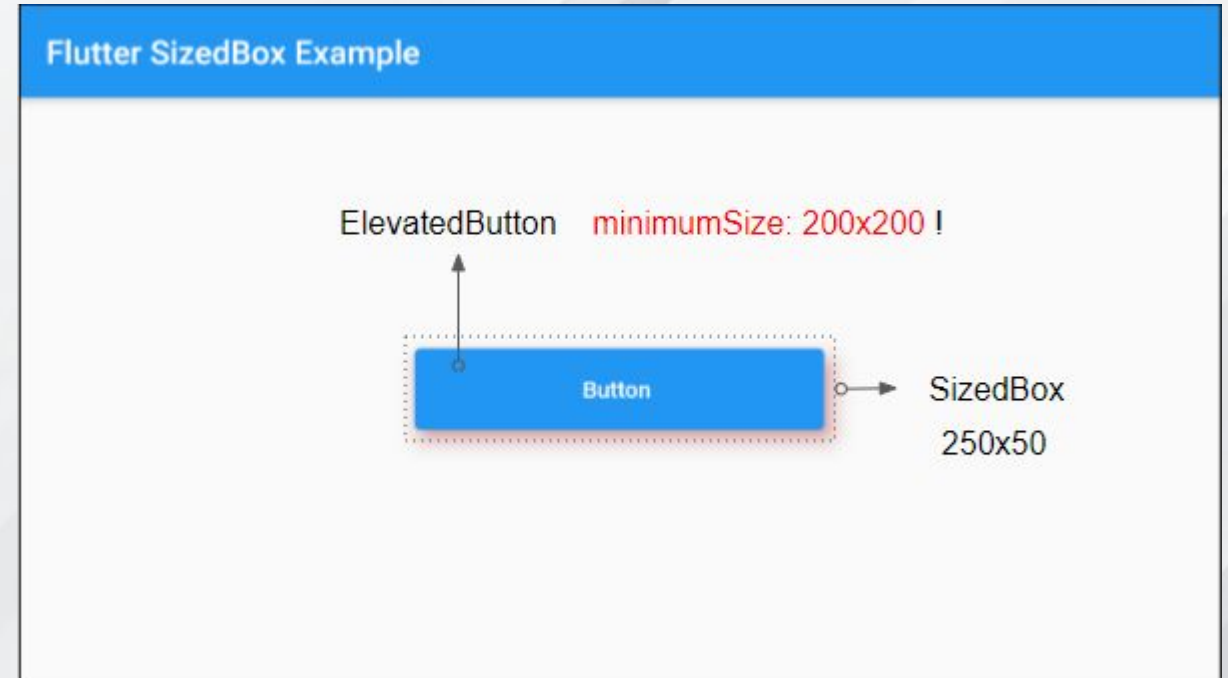
Container

- ◆ Same features for sized box
- ◆ Have Decoration
- ◆ Padding and margin features

Code Example

Sized Box:

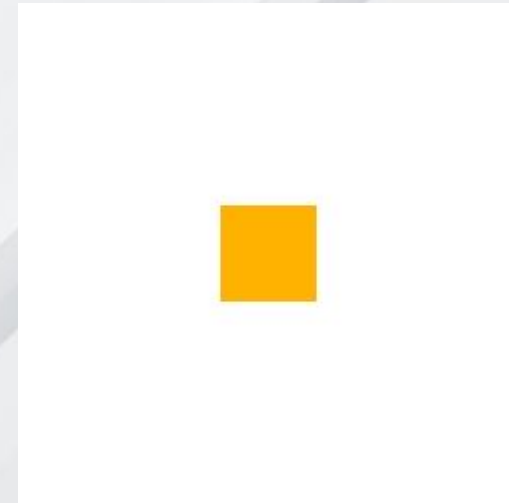
```
SizedBox(  
  height: 50,  
  width: 250,  
)
```



Code Example

Container:

```
Container(  
  margin: const EdgeInsets.all(10.0),  
  color: Colors.amber[600],  
  width: 48.0,  
  height: 48.0,  
)
```



Widgets

Text

Code Example:

```
Text( 'Hello, $_name! How are you?',  
      textAlign: TextAlign.center,  
      style: const TextStyle(fontWeight:  
        FontWeight.bold  
      ),  
    )
```

Hello, Ruth! How are you?

Gestures

It is a widget that provides interaction (how to listen for and respond to) in Flutter using GestureDetector.

GestureDetector is an invisible widget, which includes tapping, dragging, and scaling interaction of its child widget.

We can also use other interactive features into the existing widgets by composing with the GestureDetector widget.

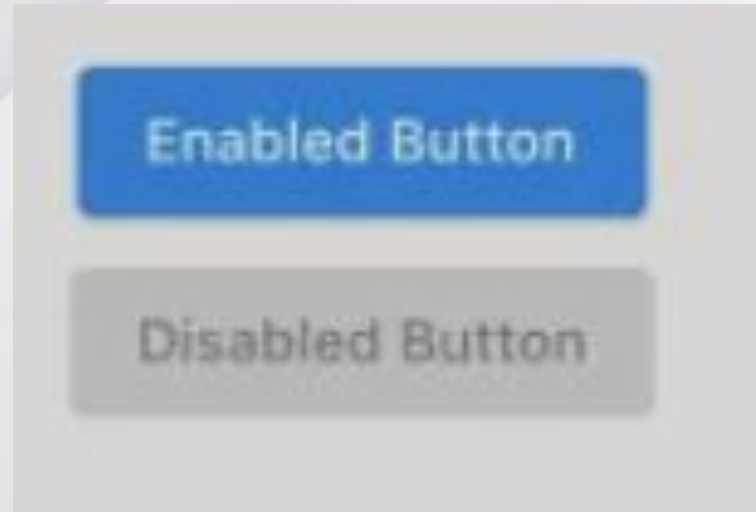
Button

Elevated Button

Code Example:

```
// An enabled button
ElevatedButton(
  child: const Text('Enabled Button'),
  onPressed: () {}, ),

// A disabled button
ElevatedButton(
  onPressed: () {},
  child: Text('Disabled Button')
)
```



Icon

This widget acts as a container for storing the Icon in the Flutter. The following code explains it more clearly.

```
new Icon(  
  Icons.add,  
  size: 34.0,  
)
```

Padding

This widget wraps other widgets to give them padding in specified directions. You can also provide padding in all directions. We can understand it from the below example that gives the text widget padding of 6.0 in all directions.

Example Padding(

```
padding: const EdgeInsets.all(6.0), child: new Text(
```

```
"Element 1", ),
```

```
),
```

Stack

It is an essential widget, which is mainly used for overlapping a widget, such as a button on a background gradient.

Widgets

Column

- Adding Elements as vertical way in a column

Row

- ◆ Adding elements in horizontal way in a row



Row

Layout a list of child widgets in the horizontal direction.



Column

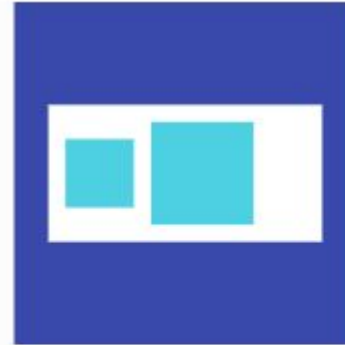
Layout a list of child widgets in the vertical direction.

Widgets

Row

Code Example:

```
Row(
  children: [
    Container(
      color: Colors.black,
      height: 20,
      width: 60,
    ),
    Container(
      color: Colors.blue,
      height: 20,
      width: 60,
    ),
  ],
)
```



Row

Layout a list of child widgets in the horizontal direction.



Column

Layout a list of child widgets in the vertical direction.

Widgets

Column

Code Example:

```
Column(
  children: [
    Container(
      color: Colors.black,
      height: 20,
      width: 60,
    ),
    Container(
      color: Colors.blue,
      height: 20,
      width: 60,
    ),
  ],
)
```



Row

Layout a list of child widgets in the horizontal direction.



Column

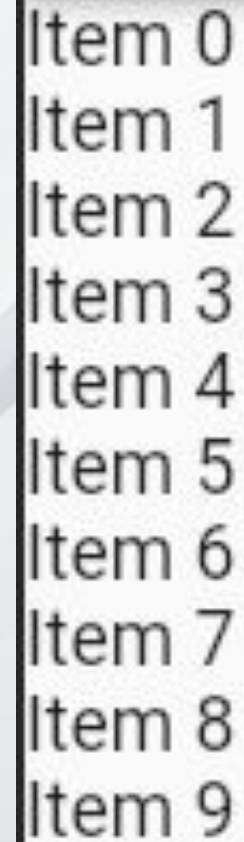
Layout a list of child widgets in the vertical direction.

Widgets

Listview

Code Example:

```
ListView.builder(  
  itemCount: 10,  
  itemBuilder: (context, index) {  
    return Text('Item $index');  
  },  
),
```



Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9

Widgets

Image.Assets

- Display images that saved in assets folder for project.

Image.network

- ◆ Display images from internet by taking URL for image.

Widgets

Image.Assets

Code Example:

```
Image.asset(  
  'assets/images/geeksforgeeks.jpg',  
  height: 400,  
  width: 400,  
),
```

Image.Network

Code Example:

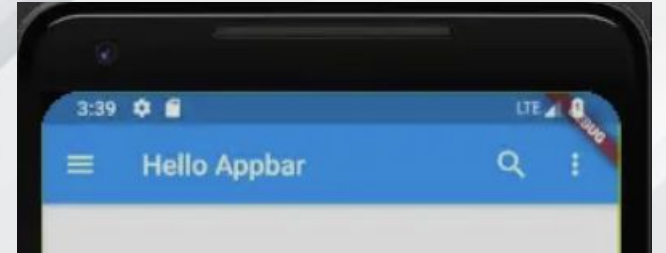
```
Image.network(  
  'https://miro.medium.com/max/839/1*KQADR  
GpUgkINaEAeCWYs8w.png',  
  height: 400,  
  width: 400,  
),
```

Widgets

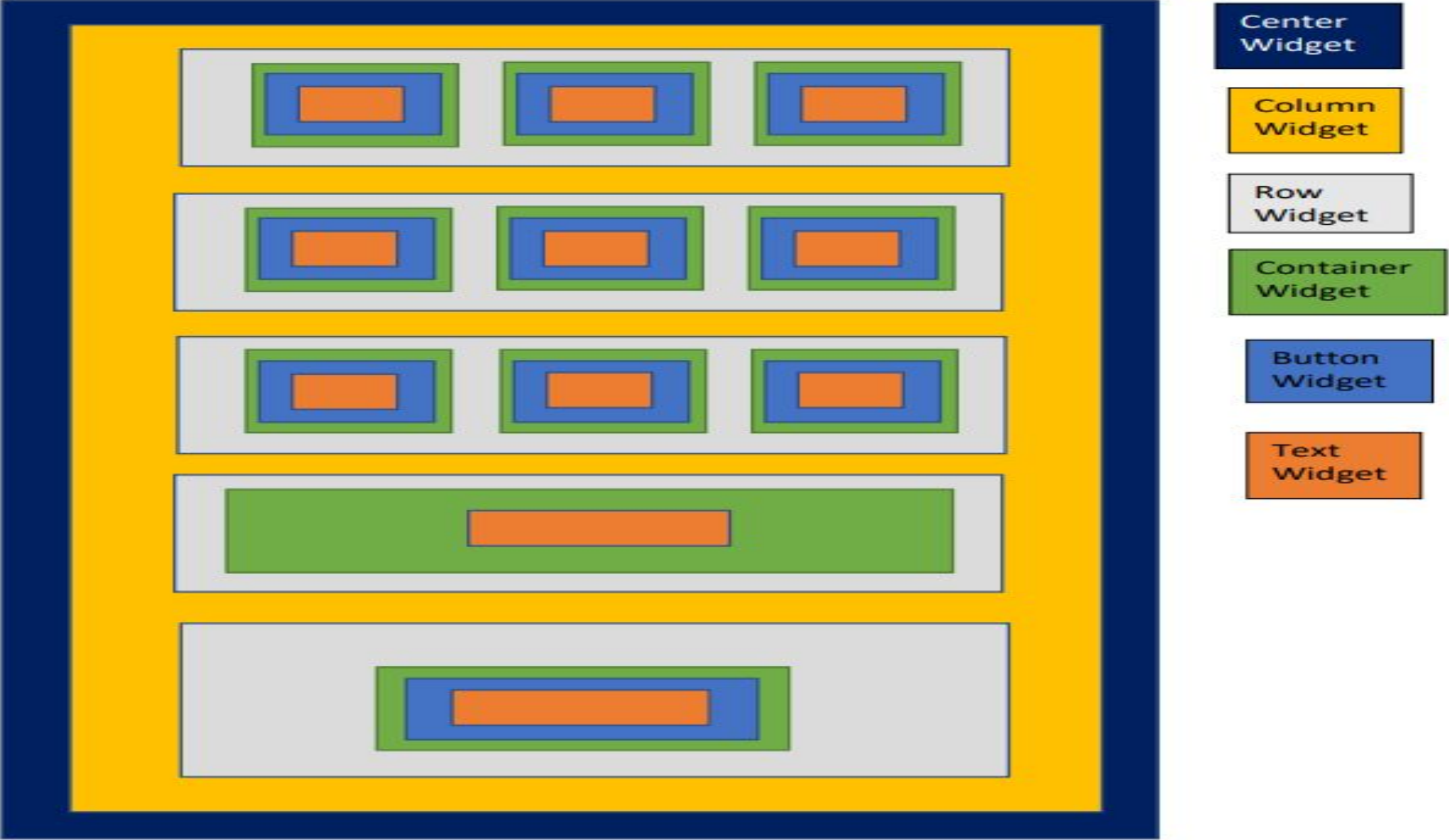
AppBar

Code Example:

```
AppBar(
  title: Text("Hello Appbar"),
  leading: GestureDetector(
    onTap: () { /* Write code here */ },
    child: Icon(
      Icons.menu,
    ),
  ),
  actions: <Widget>[
    Icon(
      Icons.search,
      size: 26.0,
    ),
    Icon(
      Icons.more_vert
    ),
  ],
),
```



Widget Tree



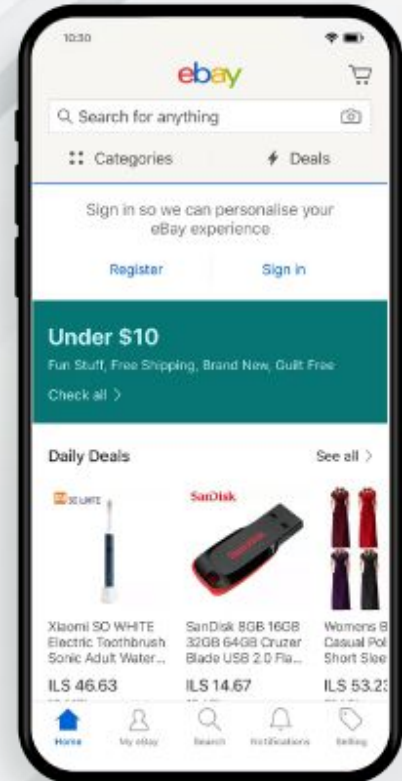
Material

Material

- ❑ MaterialApp is a widget that introduces a number of widgets Navigator, Theme that are required to build a material design app.

Code Example:

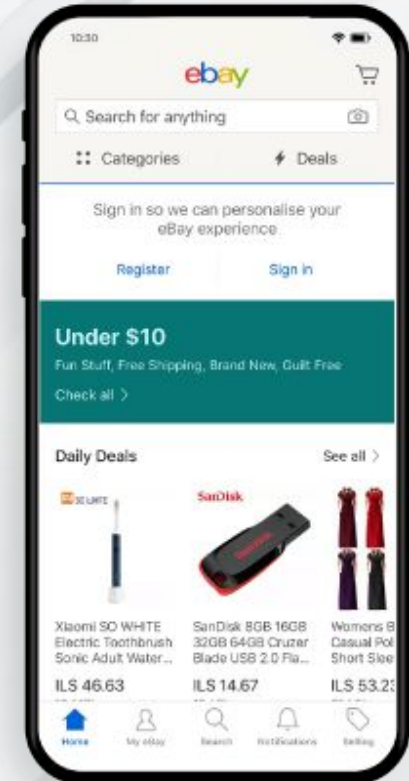
```
MaterialApp(  
  title: 'IEEE',  
  theme: ThemeData(primarySwatch: Colors.green),  
  home: Scaffold(  
    appBar: AppBar(title: const Text('IEEE')),  
  ),  
);
```



Material

To start Flutter programming, you need first to import the Flutter package.

Here, we have imported a Material package. This package allows you to create user interface according to the Material design guidelines specified by Android.



Scaffold

- ❑ is used under MaterialApp, it gives you many basic functionalities, like AppBar, BottomNavigationBar, Drawer, FloatingActionButton, etc.

Code Example:

```
Scaffold(
  appBar: AppBar(title: const Text('IEEE')),
  body: Column(
    children :[
      ]
    ),
),
```

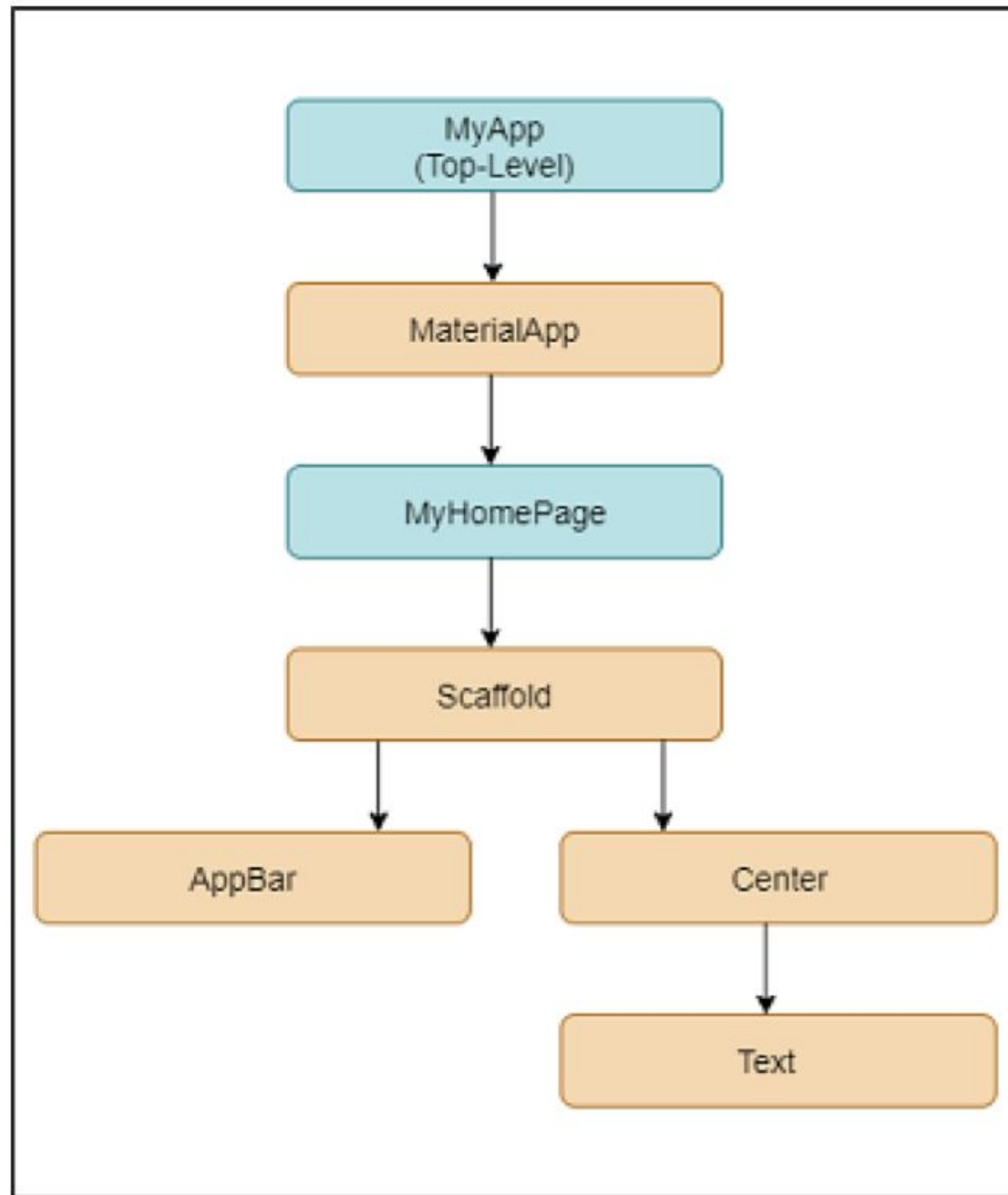


runApp

The second line is an entry point of the Flutter applications similar to the main method in other programming languages.

It calls the runApp function and pass it an object of MyApp The primary purpose of this function is to attach the given widget to the screen.

```
runApp(MyWidget());
```

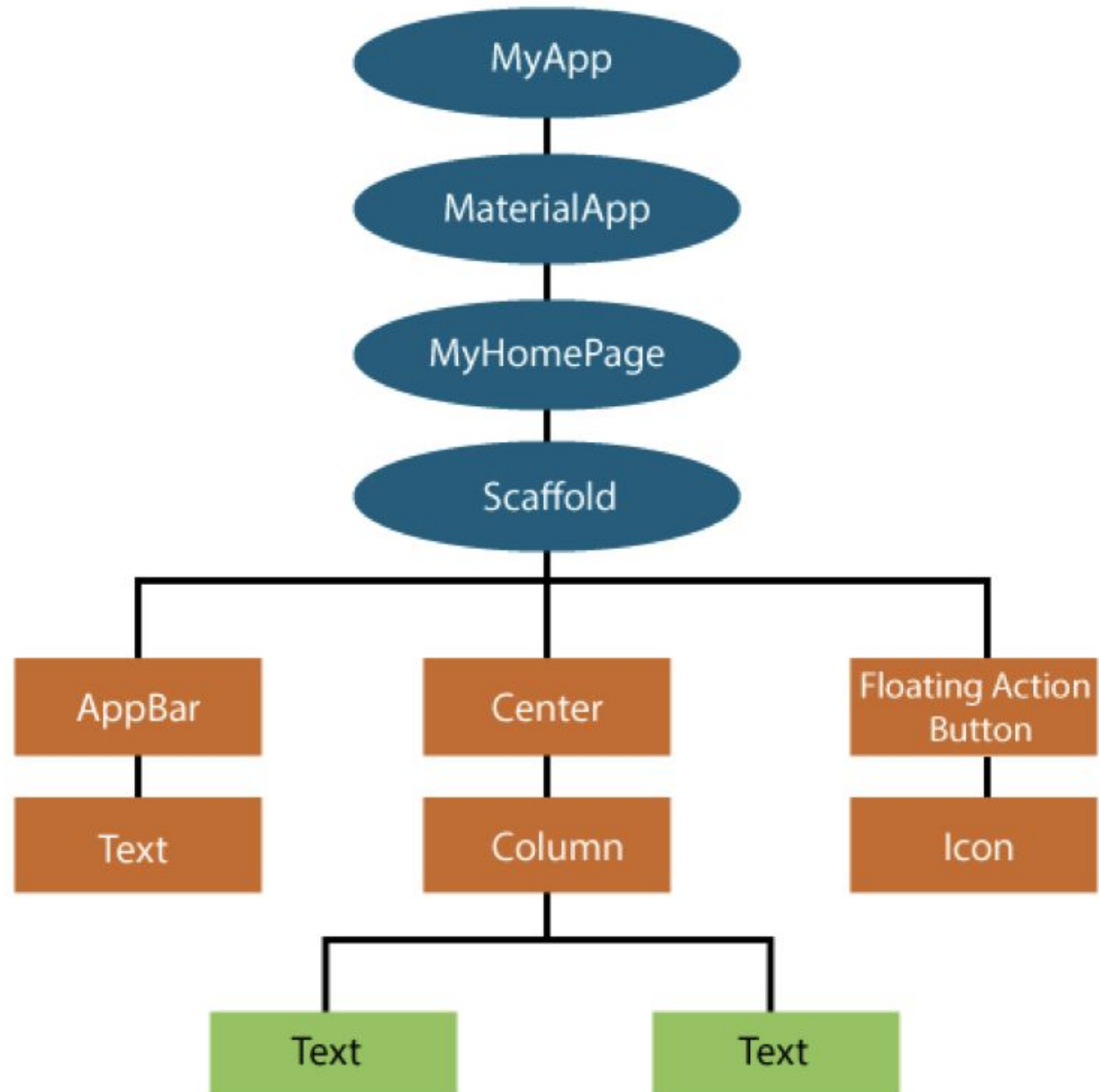


Chapter
01

My Home Page

- the MyHomePage is similar to MyApp, except it will return the Scaffold Scaffold widget is a top-level widget after the MaterialApp widget for creating the user interface.
- This widget contains two properties appBar and body. The appBar shows the header of the app, and body property shows the actual content of the application.
- Here, AppBar render the header of the application, Center widget is used to center the child widget, and Text is the final widget used to show the text content and displays in the center of the screen.





Chapter
01**Layers**

- Layers are an important concept of the Flutter framework, which are grouped into multiple categories in terms of complexity and arranged in the top-down approach.
- The topmost layer is the UI of the application, which is specific to the Android and iOS platforms. The second topmost layer contains all the Flutter native widgets.
- The next layer is the rendering layer, which renders everything in the Flutter app.
- Then, the layers go down to Gestures, foundation library, engine, and finally, core platform-specific code.



Live Coding

What is Next ?

<https://zapp.run/>



Thank You