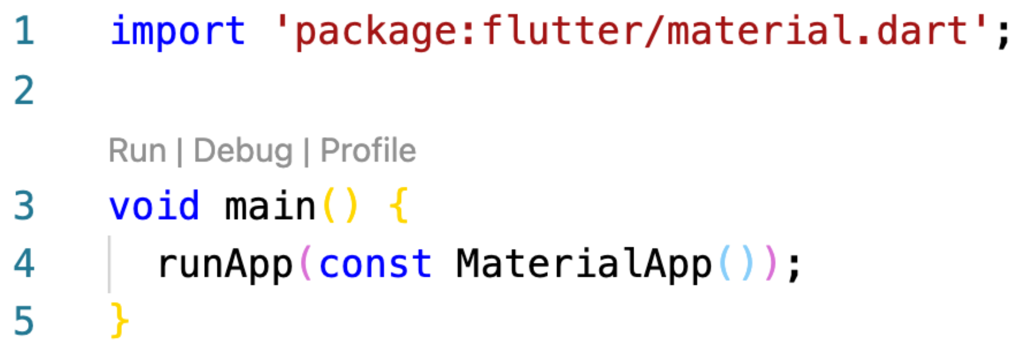
Introduction to Widgets:



```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MaterialApp());
}
```

The first thing before you start is to import the ‘material.dart’ package. To run the ‘MaterailApp’ you have to import this package.

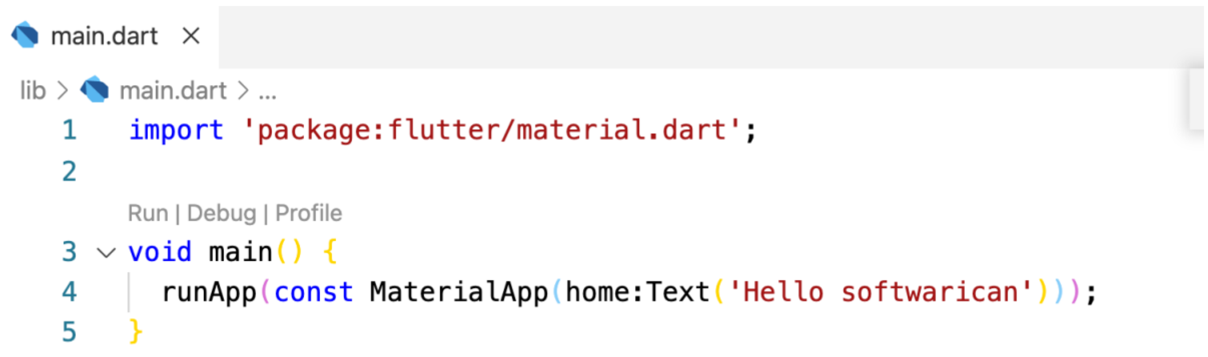
So here we are going to run the ‘MaterialApp’.
‘MaterialApp’ is a app which used the Material design pattern. Material design is a simple design or concept for building user interface.

What is material design?

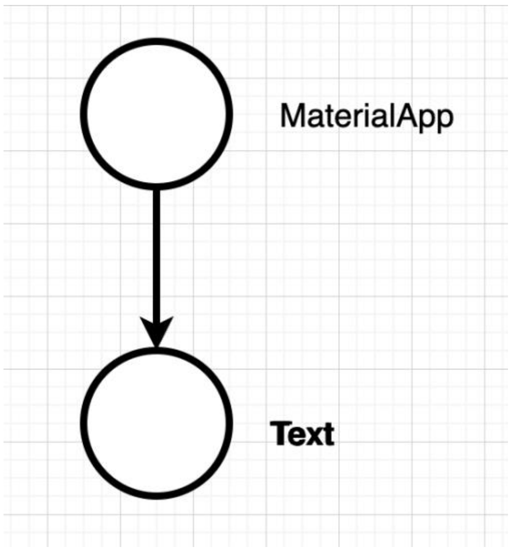
It is design language developed by Google in 2014. Expanding on the "cards" that debuted in Google Now, Material Design uses more grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows. Google announced Material Design on June 25, 2014, at the 2014 Google I/O conference.

The main purpose of Material Design is the creation of a new visual language that combines principles of good design with technical and scientific innovation. Designer Matias Duarte explained that, "unlike real paper, our digital material can expand and reform intelligently. Material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch." Google states that their new design language is based on paper and ink but implementation takes place in an advanced manner.

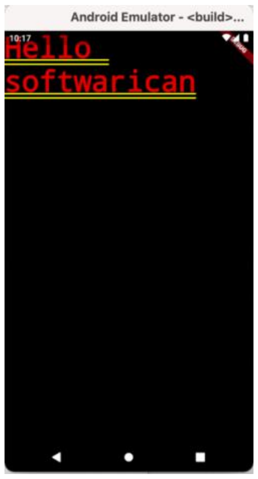
If you run the above code, then we will get the default blank material application. The most important things in above code inside the MaterialAPP() , home property which helps you add the widget in your application.



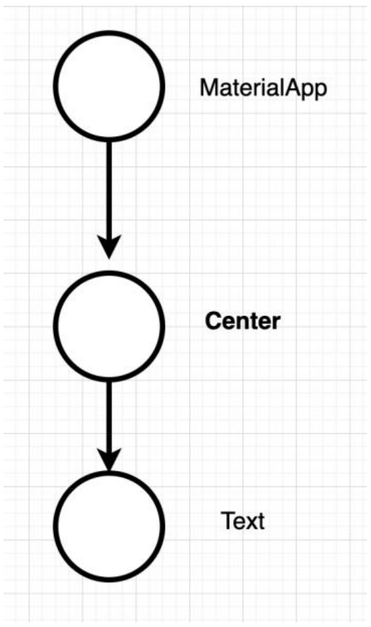
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MaterialApp(home:Text('Hello softwarican')));
}
```



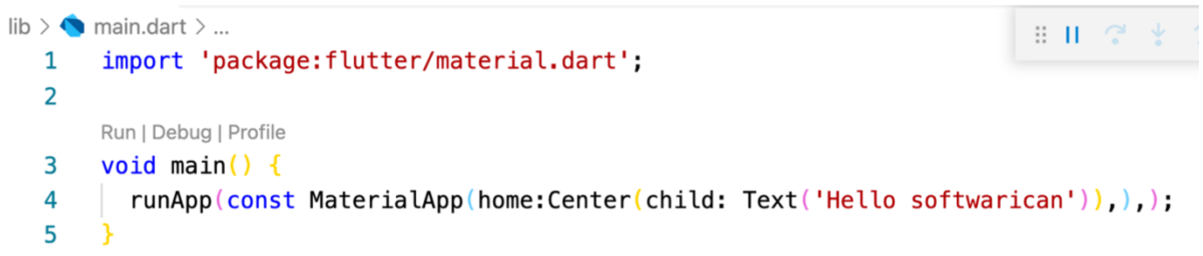
Here we have run the Material application where we have inserted the Text widget inside it. We can run this application and see the output as below:



Now how can you insert the text in the center of the screen?



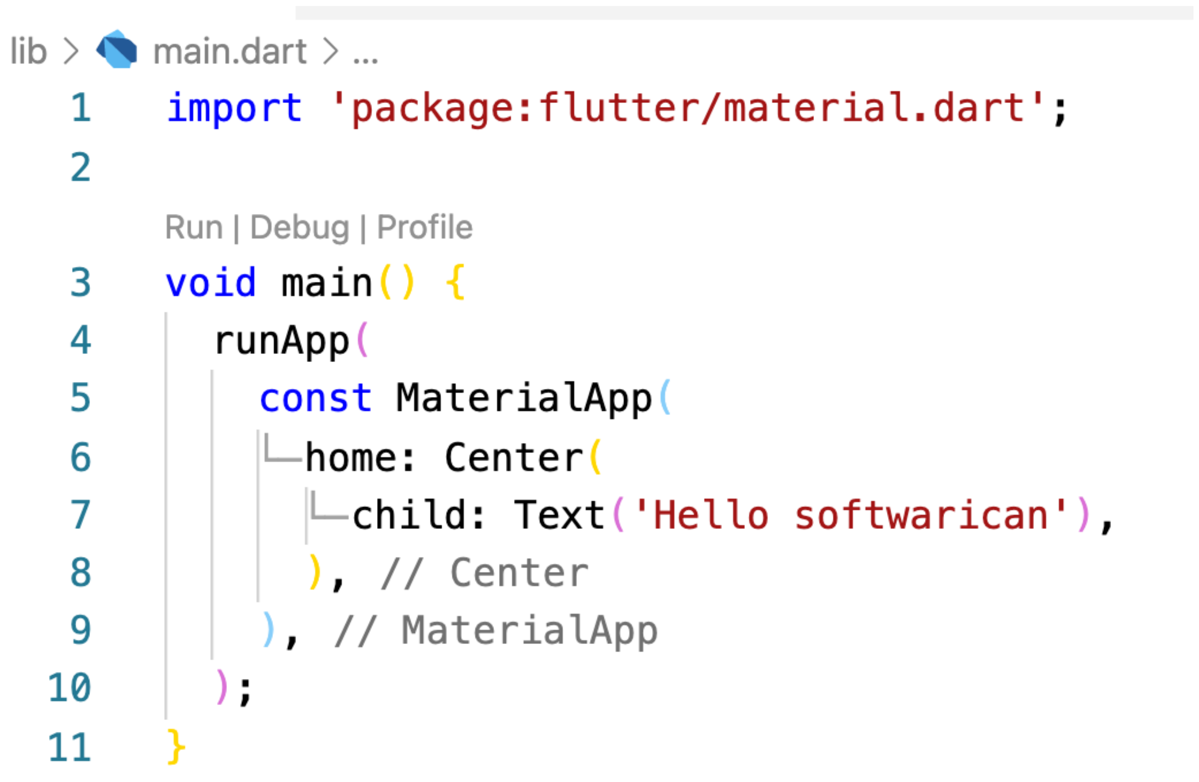
Code goes like this,



```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MaterialApp(home:Center(child: Text('Hello softwarican')),),);
}
```

As we see the above code, all the code is in same place. If we code in this manner, then it will increase the complexity of your application. So, how to make it easy?

- Flutter always recommends, when you create a widget then use comma after parenthesis or round brackets.



```dart
import 'package:flutter/material.dart';

void main() {
  runApp(
    const MaterialApp(
      home: Center(
        child: Text('Hello softwarican'),
      ), // Center
    ), // MaterialApp
  );
}
```

Up to here, we have created a new Material App. By including Material App, we can use all the components or widgets that come up with 'MaterialApp'.

Scaffold () Widget:

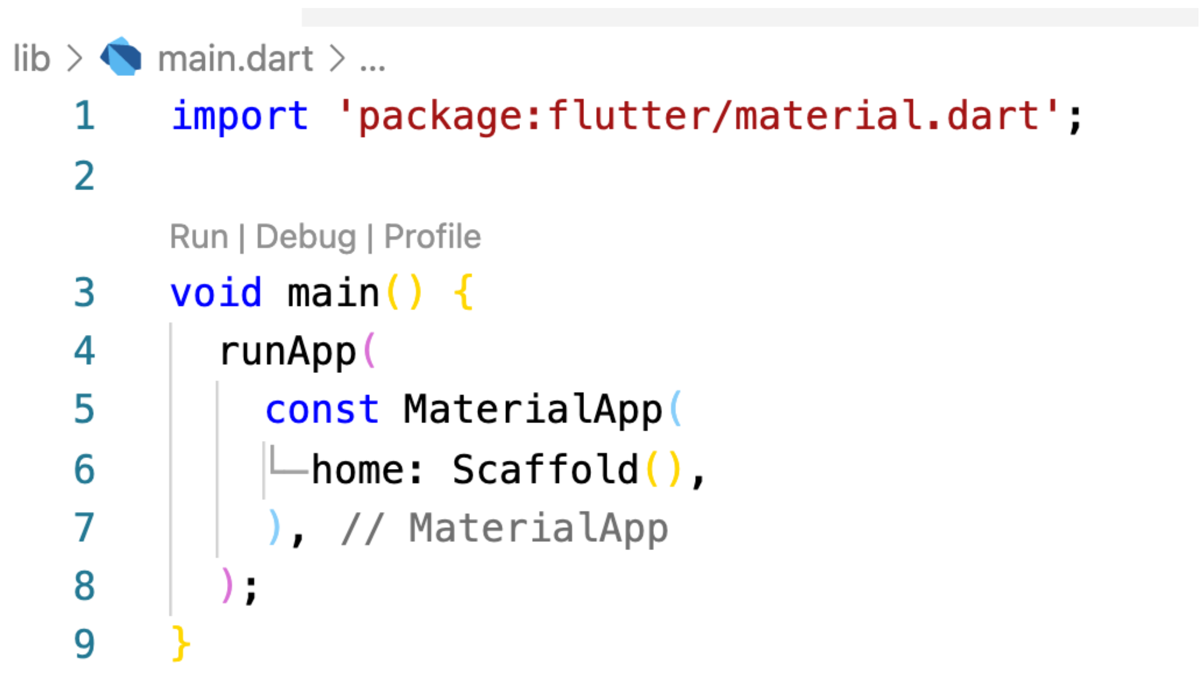
The Scaffold is a widget in Flutter used to implements the basic material **design visual layout structure**. It is quick enough to create a general-purpose mobile application and contains almost everything we need to create a functional and responsive Flutter apps. This widget is able to occupy the whole device screen. In other words, we can say that it is mainly responsible for creating a base to the app screen on which the child widgets hold on and render on the screen. It provides many widgets or APIs for showing Drawer, SnackBar, BottomNavigationBar, AppBar, FloatingActionButton, and many more.

The Scaffold class is a shortcut to set up the look and design of our app that allows us not to build the individual visual elements manually. It saves our time to write more code for the look and feel of the app. The following are the **constructor and properties** of the Scaffold widget class.

1. const Scaffold({

```
Key key,
this.appBar,
this.body,
this.floatingActionButton,
this.floatingActionButtonLocation,
this.persistentFooterButtons,
this.drawer,
this.endDrawer,
this.bottomNavigationBar,
this.bottomSheet,
this.floatingActionButtonAnimator,
this.backgroundColor,
this.resizeToAvoidBottomPadding = true,
this.primary = true,
})
```

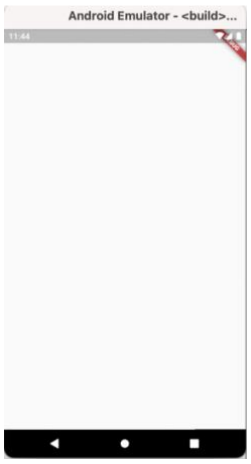
Let’s change the above code, instead of using other widget in home properties of MaterailApp we are going to use Scaffold() widget.



lib > main.dart > ...

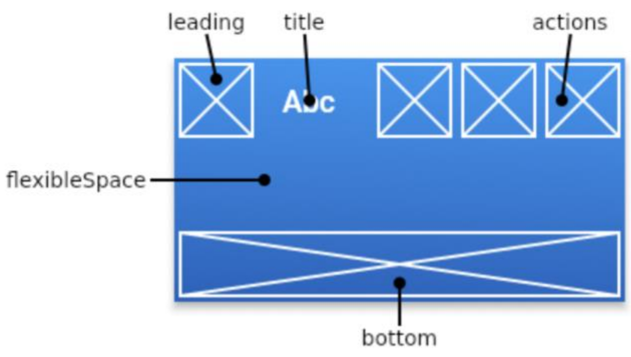
```
import 'package:flutter/material.dart';

Run | Debug | Profile
void main() {
  runApp(
    const MaterialApp(
      home: Scaffold(),
    ), // MaterialApp
  );
}
```

If you run this above code, we will see some changes in the output:



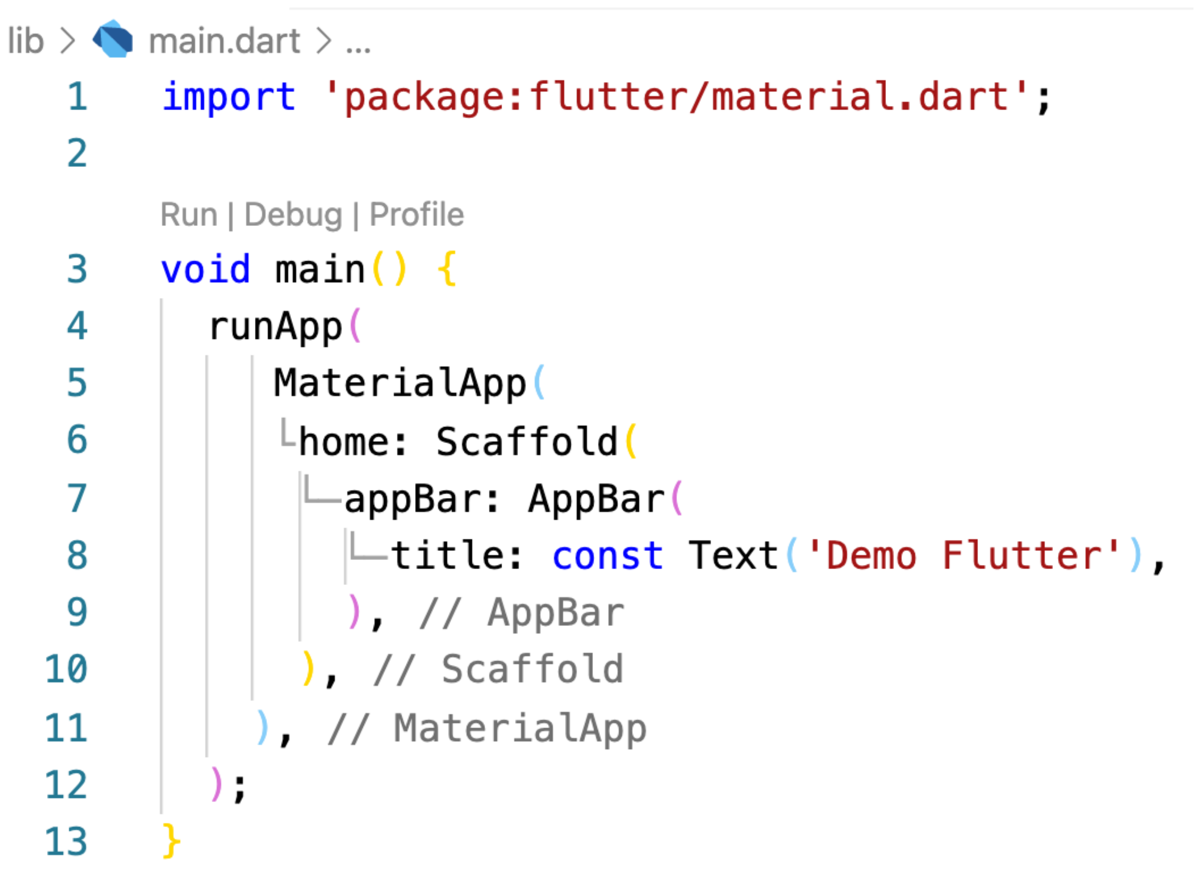
Let us understand all of the above properties in detail.

appBar: It is a **horizontal bar** that is mainly displayed at the **top** of the Scaffold widget. It is the main part of the Scaffold widget and displays at the top of the screen. Without this property, the Scaffold widget is incomplete. It uses the appBar widget that itself contains various properties like elevation, title, brightness, etc. See the below example:



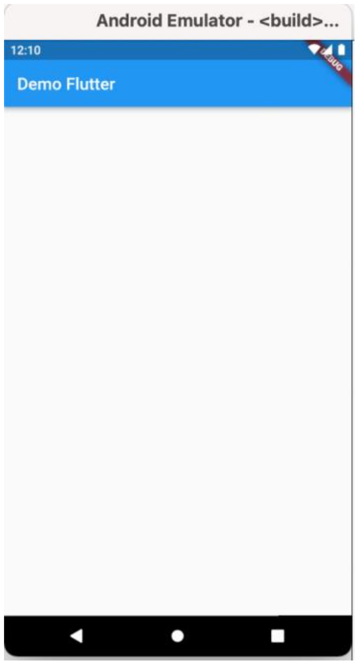
'appBar' properties will take an 'AppBar()' widget which has again different properties:

AppBar() widget helps us to add title, leading, button, backgroundColor and many more.



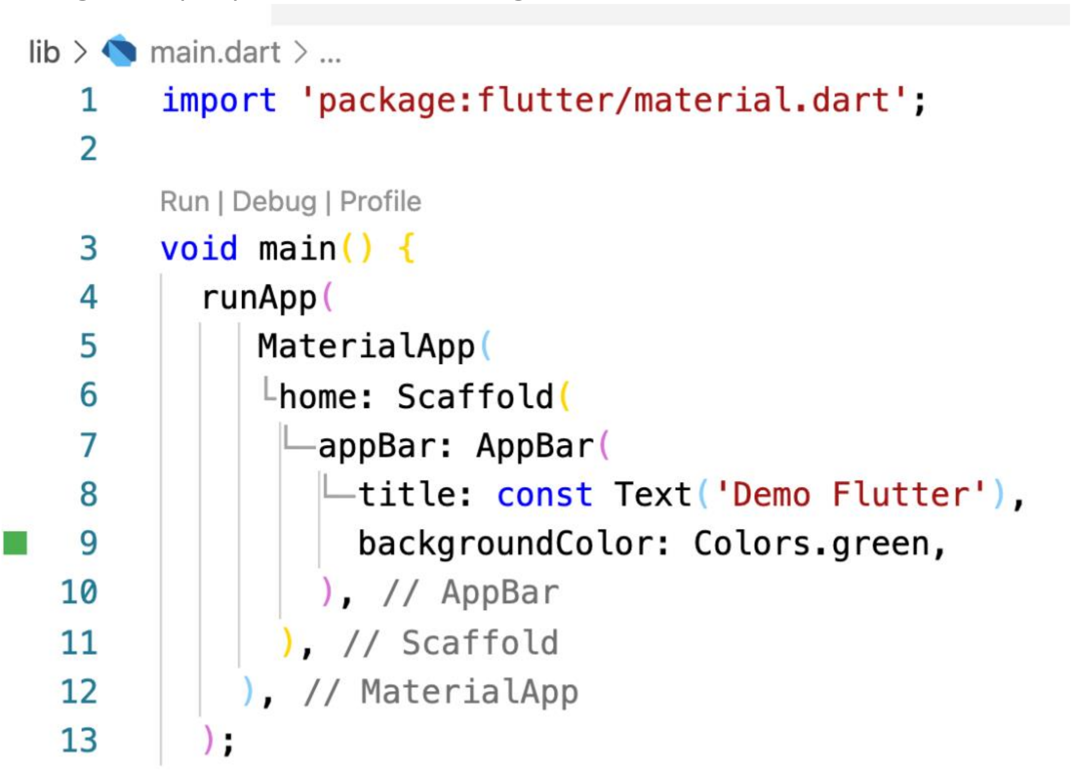
lib > main.dart > ...

```dart
import 'package:flutter/material.dart';

Run | Debug | Profile
void main() {
  runApp(
    MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Demo Flutter'),
        ), // AppBar
      ), // Scaffold
    ), // MaterialApp
  );
}
```



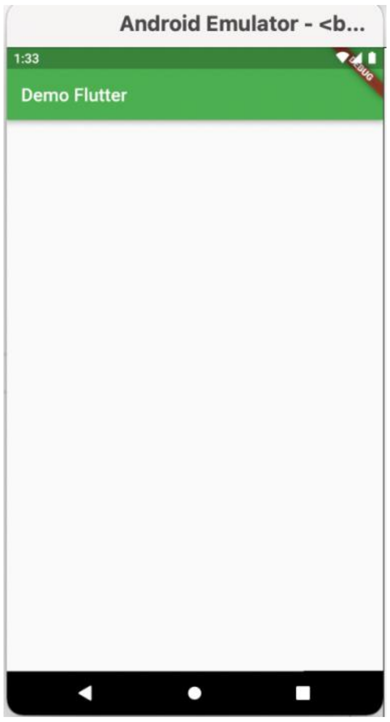
Just then I want to change the background color of appbar, but not of scaffold.

Now, just remember the things, which widgets you are going to change, change the properties of that widget.


lib > main.dart > ...

```dart
import 'package:flutter/material.dart';

Run | Debug | Profile
void main() {
  runApp(
    MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: const Text('Demo Flutter'),
        backgroundColor: Colors.green,
      ), // AppBar
    ), // Scaffold
  ), // MaterialApp
  );
```

Now if you run this code, you will see the changes as below:

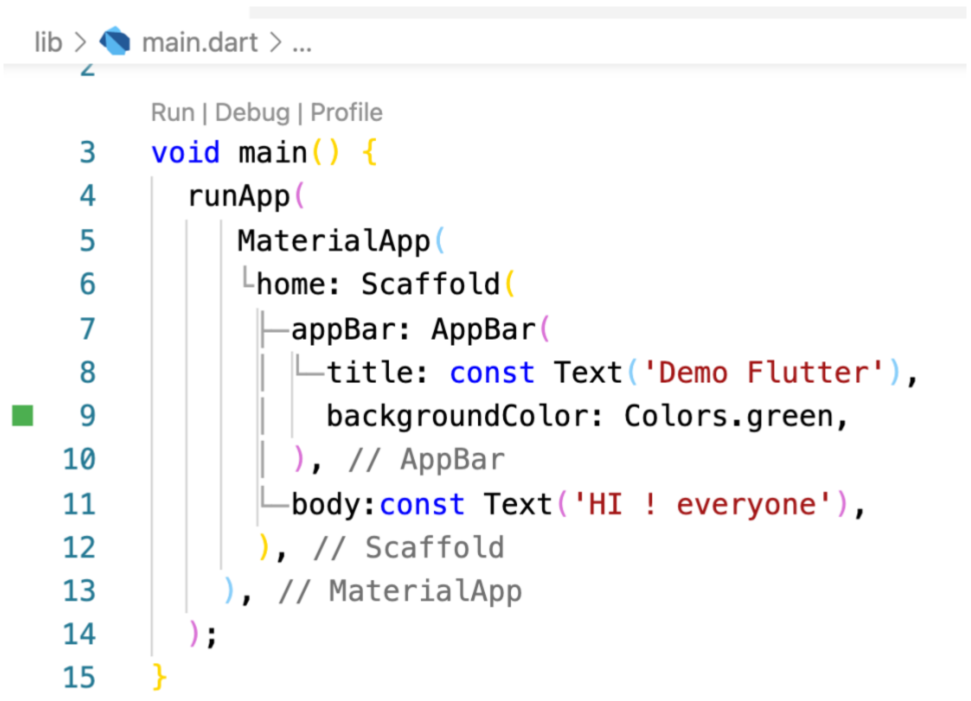


If you want to change the background color of Scaffold, you must change or include the properties of Scaffold.

Now if you want to include some changes in the body of Scaffold then what will you do?

To change or include some widget in body of Scaffold you must add a body property of a Scaffold.

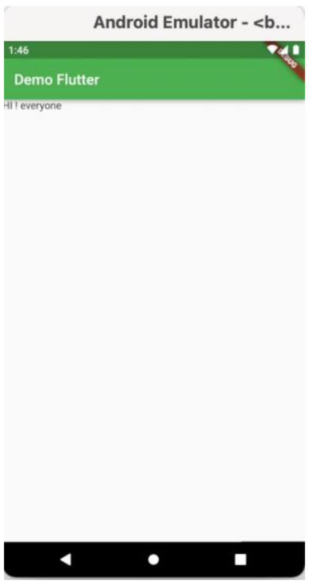
body: It is the other primary and required property of this widget, which will **display the main content in the Scaffold**. It signifies the place below the appBar and behind the floatingActionButton & drawer. The widgets inside the body are positioned at the top-left of the available space by default. See the below code:



lib > main.dart > ...

Run | Debug | Profile

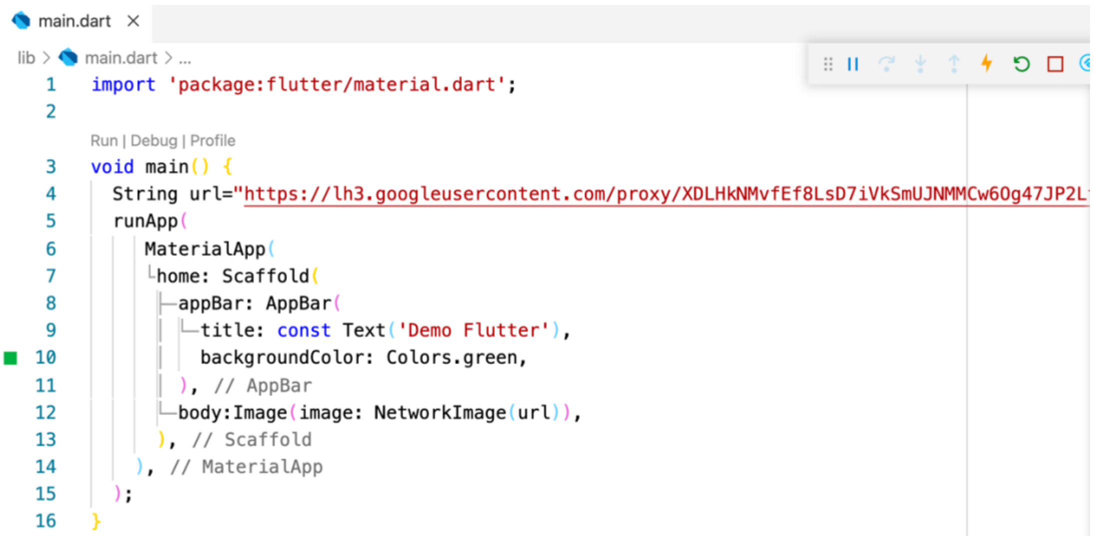
```
void main() {
  runApp(
    MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: const Text('Demo Flutter'),
        backgroundColor: Colors.green,
      ), // AppBar
      body:const Text('HI ! everyone'),
    ), // Scaffold
   ), // MaterialApp
  );
}
```



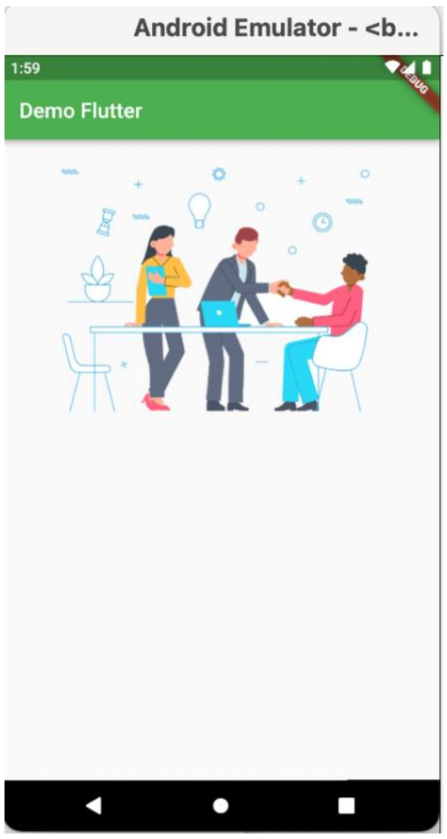
As we have discussed earlier, to make the text in the center you must use the Center Widget to put the text in the center of the body of the Scaffold.

Task: Please put the above text in the center of screen.

Not only text, but we can also add any widget in the body of the scaffold. Let us add the image in the body of screen or scaffold.

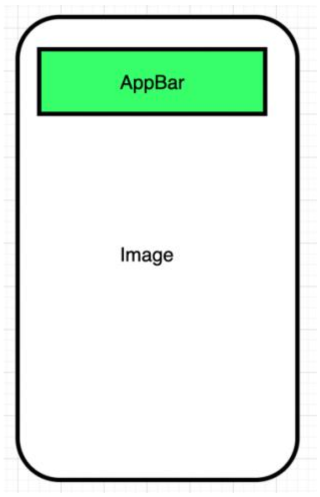


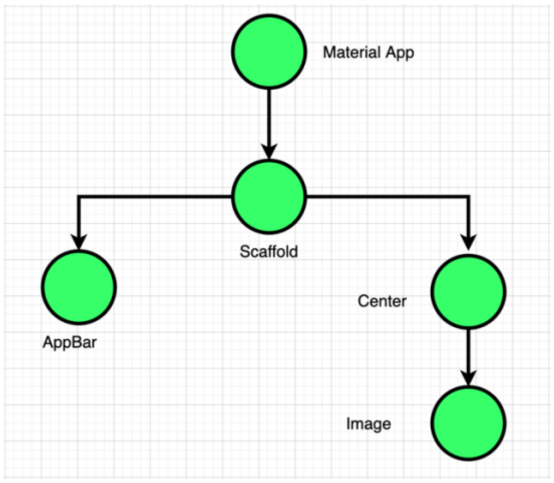
```
import 'package:flutter/material.dart';

void main() {
  String url="https://lh3.googleusercontent.com/proxy/XDLHkNMvfEf8LsD7iVkSmUJNMMCw6Og47JP2L
  runApp(
     MaterialApp(
     home: Scaffold(
       appBar: AppBar(
         title: const Text('Demo Flutter'),
         backgroundColor: Colors.green,
       ), // AppBar
       body:Image(image: NetworkImage(url)),
     ), // Scaffold
    ), // MaterialApp
  );
}
```



Task: Put this image in the center of the screen.

Hints:





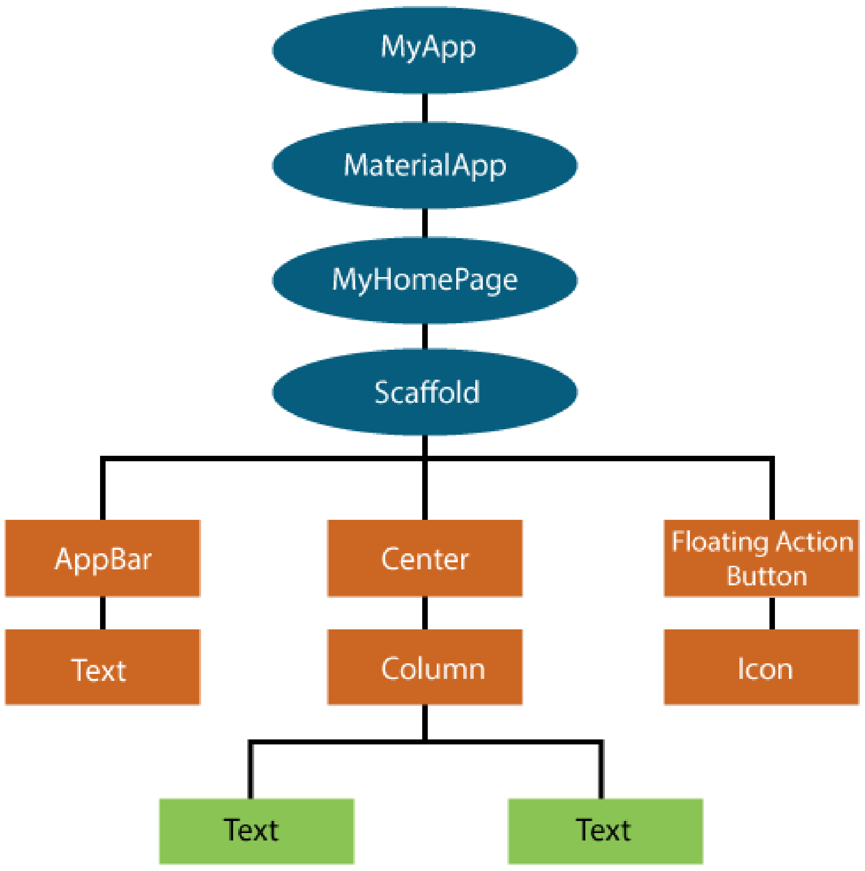
Flutter Widgets:

In this section, we are going to learn the concept of a widget, how to create it, and their different types available in the Flutter framework. We have learned earlier that everything in Flutter is a widget.

Whenever you are going to code for building anything in Flutter, it will be inside a widget. The central purpose is to build the app out of widgets. It describes how your app view should look like with their current configuration and state. When you made any alteration in the code, the widget rebuilds its description by calculating the difference of previous and current widget to determine the minimal changes for rendering in UI of the app.

Widgets are nested with each other to build the app. It means the root of your app is itself a widget, and all the way down is a widget also. For example, a widget can display something, can define design, can handle interaction, etc.

The below image is a simple visual representation of the widget tree.



We can create the Flutter widget like this:

```
Class ImageWidget extends StatelessWidget {
        // Class Stuff
}
```

Hello World Example

```
import 'package:flutter/material.dart';

class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  // This widget is the home page of your application.
  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(this.title),
      ),
      body: Center(
        child: Text('Hello World'),
      ),
    );
  }
}
```

Types of Widget

We can split the Flutter widget into two categories:

1. Visible (Output and Input)
2. Invisible (Layout and Control)

Visible widget

The visible widgets are related to the user input and output data. Some of the important types of this widget are:

- Text
- Image
- Icon
- Button etc.

Invisible widget

The invisible widgets are related to the layout and control of widgets. It provides controlling how the widgets actually behave and how they will look onto the screen. Some of the important types of these widgets are:

- Column
- Row
- Center
- Padding
- Scaffold
- Stack etc.

State Management Widget

In Flutter, there are mainly two types of widget:

- StatelessWidget
- StatefulWidget

StatefulWidget

A StatefulWidget has state information. It contains mainly two classes: the **state object** and the **widget**. It is dynamic because it can change the inner data during the widget lifetime. This widget does not have a **build()** method. It has **createState()** method, which returns a class that extends the Flutters State Class. The examples of the StatefulWidget are Checkbox, Radio, Slider, InkWell, Form, and TextField.

Example

```
class Car extends StatefulWidget {
  const Car({ Key key, this.title }) : super(key: key);

  @override
  _CarState createState() => _CarState();
}

class _CarState extends State<Car> {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: const Color(0xFEEFE),
           child: Container(
            child: Container( //child: Container() )
        )
    );
  }
}
```

StatelessWidget

The StatelessWidget does not have any state information. It remains static throughout its lifecycle. The examples of the StatelessWidget are Text, Row, Column, Container, etc.

Example

```
class MyStatelessCarWidget extends StatelessWidget {
  const MyStatelessCarWidget ({ Key key }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(color: const Color(0x0xFEEFE));
  }
}
```