* runApp(MyApp())

void runapp(Widget app)

Takes a widget as argument, MyApp() in this case, and runs the app

* MyApp()- Calls build method
* State- the state of the widget is the information of the objects that its properties are holding at the time of its creation
* StatelessWidget – Widgets that cannot change their state dynamically, ie widgets do not get redrawn during runtime of the app. UI gets re-rendered only when input data changes and not when local state changes.

class MyApp extends StatelessWidget {

//…

}

* StatefulWidget- Widget that can change their state dynamically when user interacts during runtime of the app, ie UI gets re-rendered when either input data or local state changes.

It consists of two classes MyApp and MyAppState

class MyApp extends StatefulWidget {         /\* This class gets recreated when external

                                              data changes \*/

  @override

  State<StatefulWidget> createState() {

    // TODO: implement createState

    return \_MyAppState();

  }

}

class \_MyAppState extends State<MyApp> {     /\* The data inside this class

                                                does not get rebuilt \*/

  var \_questionIndex = 0;

  void \_answerQuestion() {

    setState(() //Calls build again and re renders UI {

      \_questionIndex = \_questionIndex + 1; //contains data which is to be altered, in

    }); this case it is \_questionIndex

    print(\_questionIndex);

  }

//…

}

* build method

The build method is called any time when setState is called, when widget's **dependencies** update, or any of the parent widgets (like Materialapp, Scaffold) are rebuilt.

[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) build(

[BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context

)

Widget : return type of the build method

* BuildContext:

BuildContext is the object type and context is the object here

BuildContext contains information about the location in the tree at which this widget is being built. BuildContext objects are passed to WidgetBuilder functions in this case it is StatelessWidget.build

* MaterialApp

A class

Wraps widgets for material design application

Takes arguments for the MaterialApp constructor

A Material app starts with the [MaterialApp](https://api.flutter.dev/flutter/material/MaterialApp-class.html) widget, which builds a number of useful widgets at the root of your app, including a [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html)(helps in smooth transition between screens of application)

Properties:

color

darkTheme

home- This is the route that is displayed first when the application is started normally

title

theme

return MaterialApp(

      home: Scaffold(

        appBar: AppBar(

          title: Text('My First App'),

        ),

        body: Column(

          children: [

//…

],

        ),

      ),

);

* Scaffold

Takes a number of different widgets as named arguments

Occupies the available space ie the whole screen.

Provide a [framework](https://www.geeksforgeeks.org/software-framework-vs-library/) to implement the basic material design layout of the application.

Properties

appBar- horizontal bar displayed at the top of the Scaffold widget

backgroundColor

body- displays main content in Scaffold. Signifies place below appBar and the widgets inside the body are positioned at the top-left of the available space by default.

* AppBar

AppBar is also a built-in class or widget in flutter.

App bars are typically used in the [Scaffold.appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) property, which places the app bar as a fixed-height widget at the top of the screen

* SilverAppBarwidget

gives scrollable functionality to the app bar.

Properties of AppBar

actions

title

backgroundColor

elevation

shape

centerTitle

bottomOpacity

foregroundColor

* Text

The [Text](https://api.flutter.dev/flutter/widgets/Text-class.html) widget displays a string of text with single style. The string might break across multiple lines or might all be displayed on the same line depending on the layout constraints.

Properties

data- text to display

style

textAlign