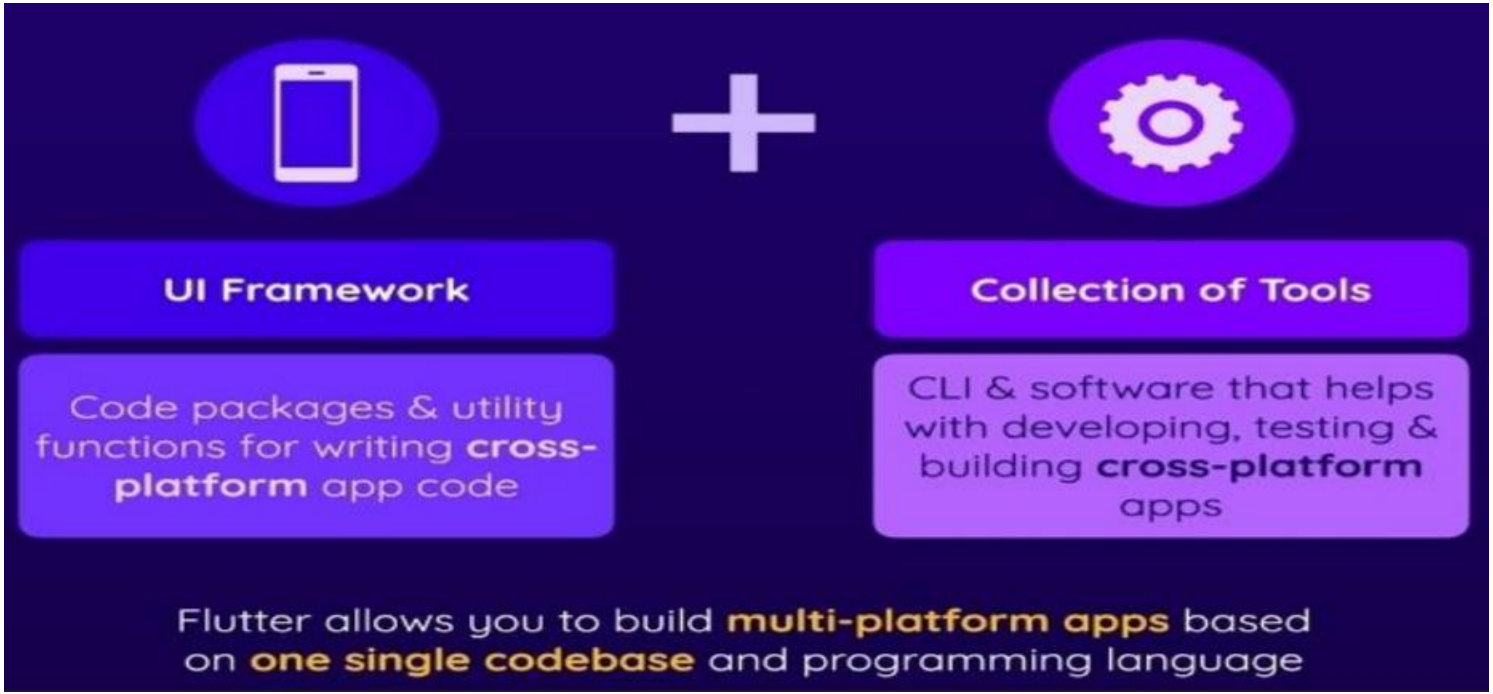
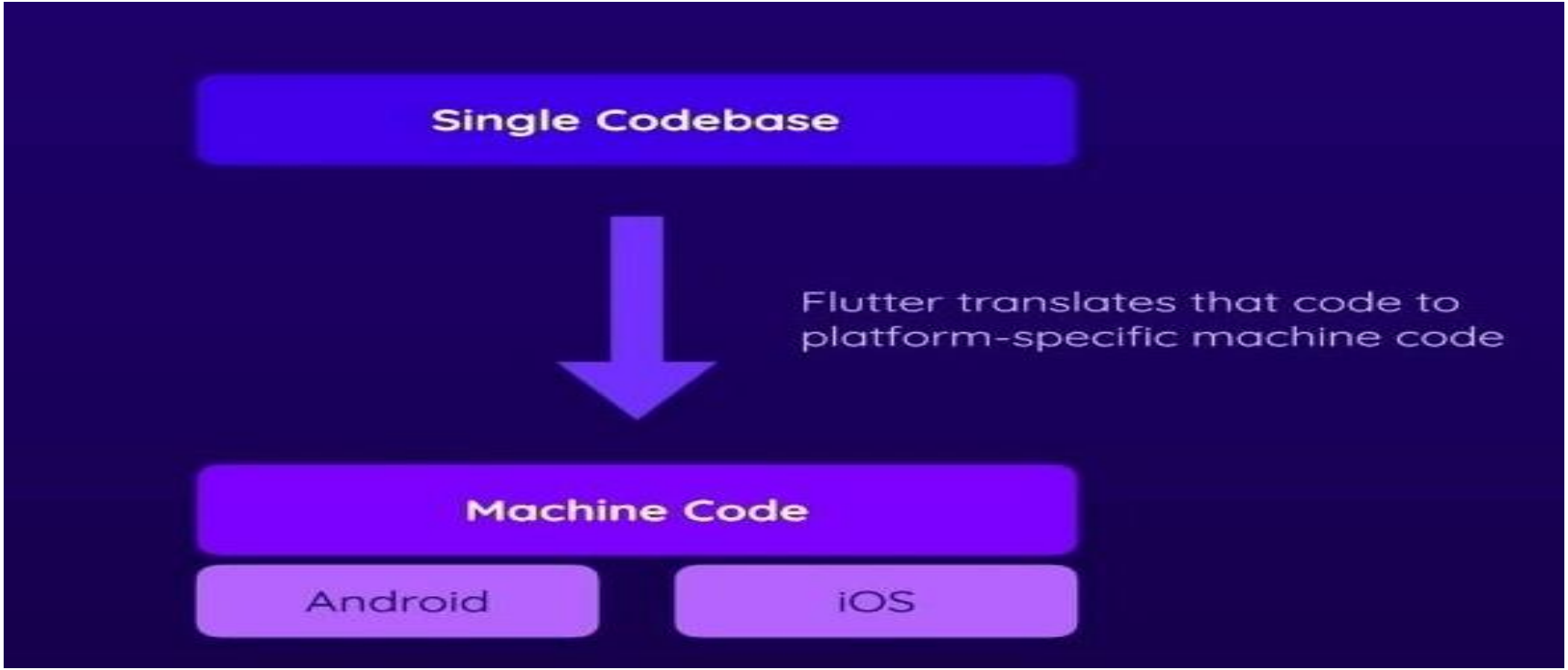
Lecture 1: Mobile Programming using Flutter

Dr.Taraggy Ghanim

What is Flutter?



From Flutter Code to Plateform Code



Flutter Is Not A Programming Language!

It's a framework for building user interfaces with Dart

Framework

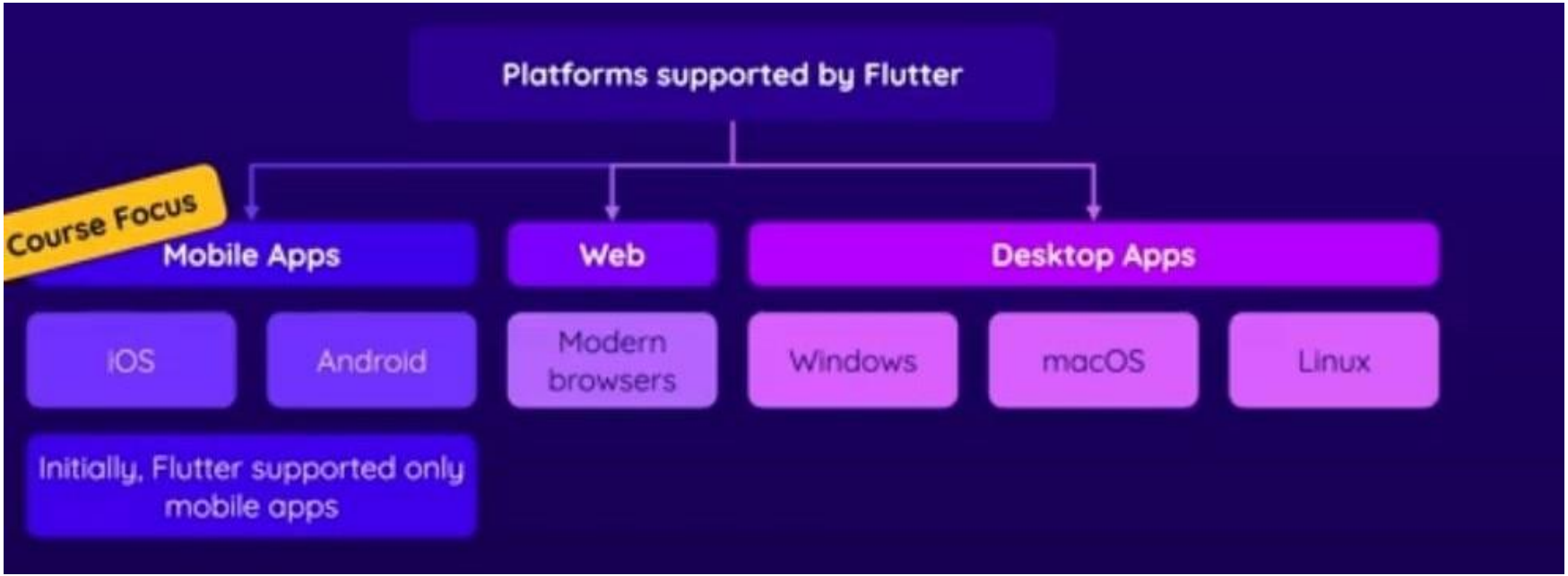
A collection of packages & utility functions you may use in your code

Dart

A programming language developed by Google

Main usage: Flutter app development

Platforms Supported by Flutter



Target Platforms

You can write the code of all platforms on the same machine.

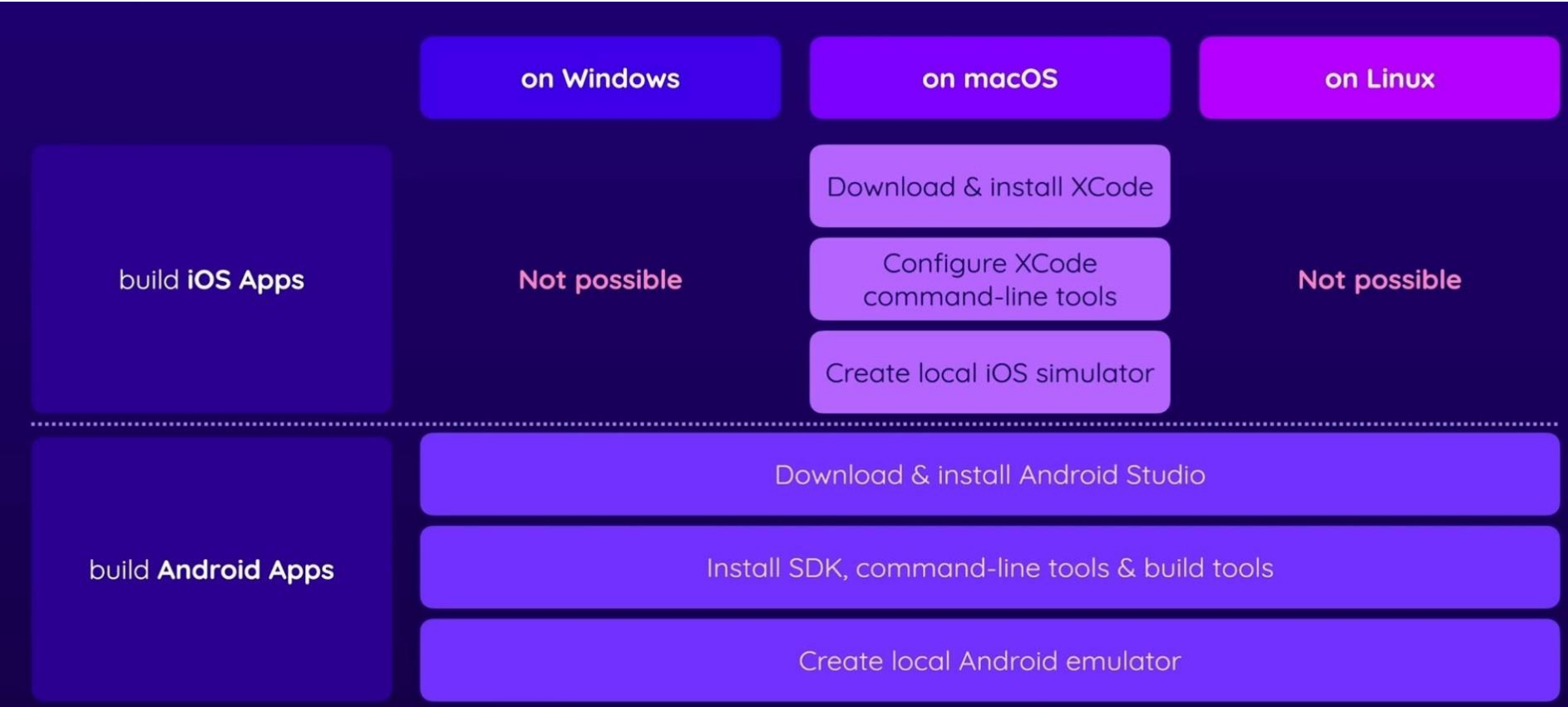
You can only test and run ios & macos on macos machine.

You can only test and run Windows app on Windows machine.

You can only test and run Linux app on Linux machine.

Android and Web apps can be built on any machine

Target Platform Tools &Devices Setup



Flutter Setup

1. **Flutter SDK**
   - Flutter SDK — For managing Flutter projects
   - Git — Version control software, used internally by Flutter SDK
2. **Platform Tools**
   - Android Studio — Used by Flutter SDK & needed for Android app deployment
   - XCode — Used by Flutter SDK & needed for iOS app deployment
3. **Virtual Devices**
   - Android — Preview Flutter apps on virtual Android devices
   - iOS — Preview Flutter apps on virtual iOS devices

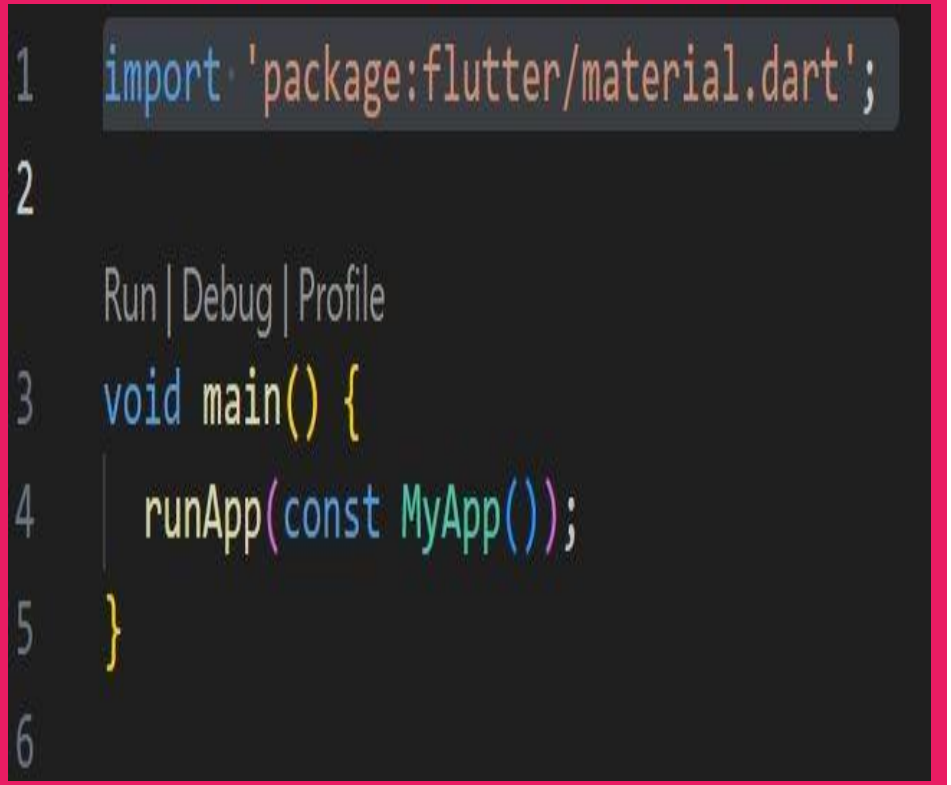
Important links

https://docs.flutter.dev/get-started/install/windows

https://github.com/academind/flutter-complete-guide-course-resources/tree/main/Lecture%20Attachments

https://github.com/academind/flutter-complete-guide-course-resources

Starting our First Program




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

`material.dart` library contains a set of pre-built widgets that implement the Material Design guidelines.

Material Design is an open-source design system built and supported by Google designers and developers.

main() runApp() functions

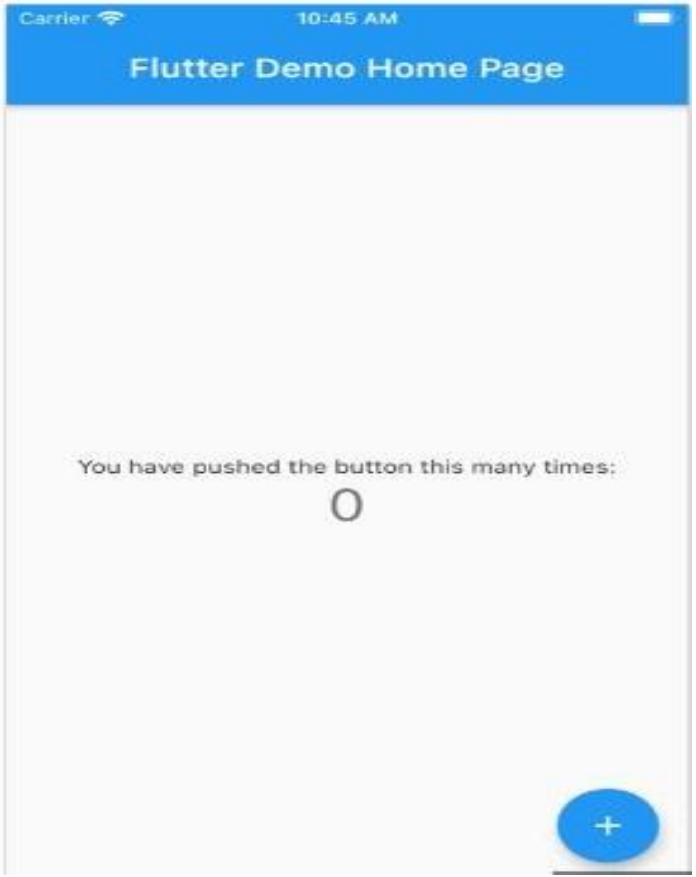
the main() function is used to start the program.

runApp() function is used to return the widgets that are connected to the screen as the root of the widget tree to be rendered on the screen.

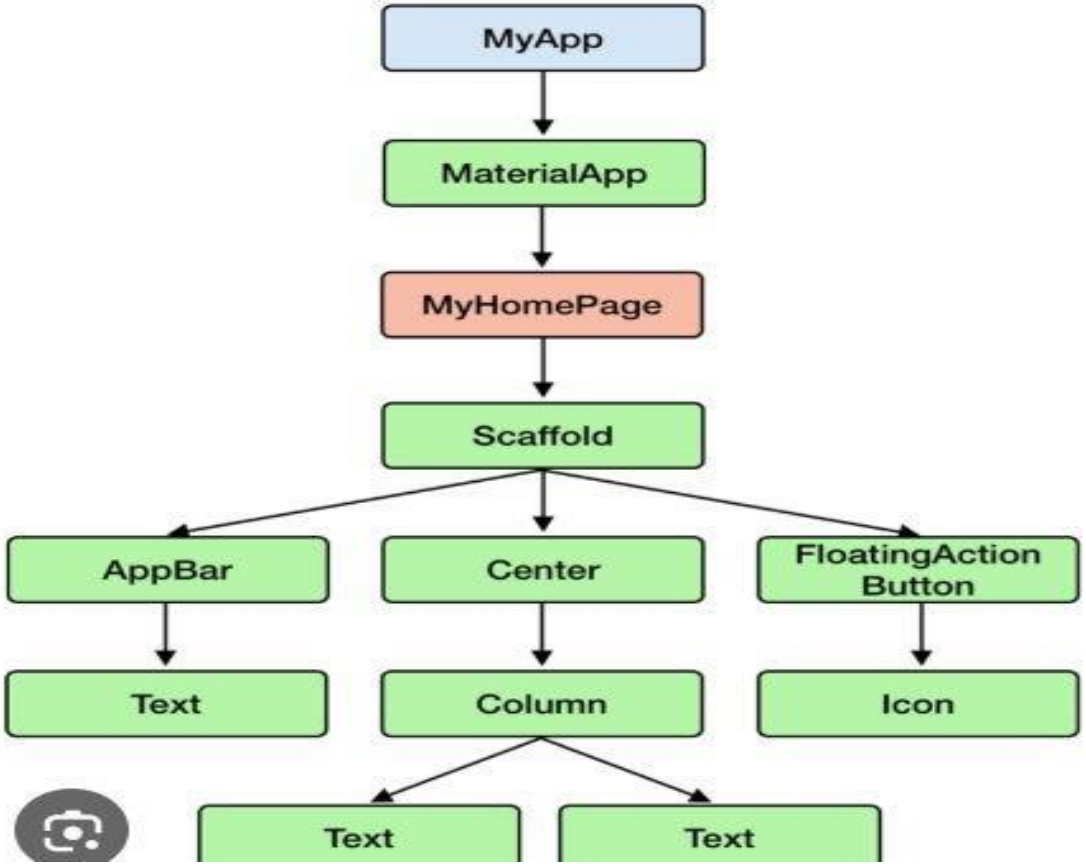
What are Widgets ?

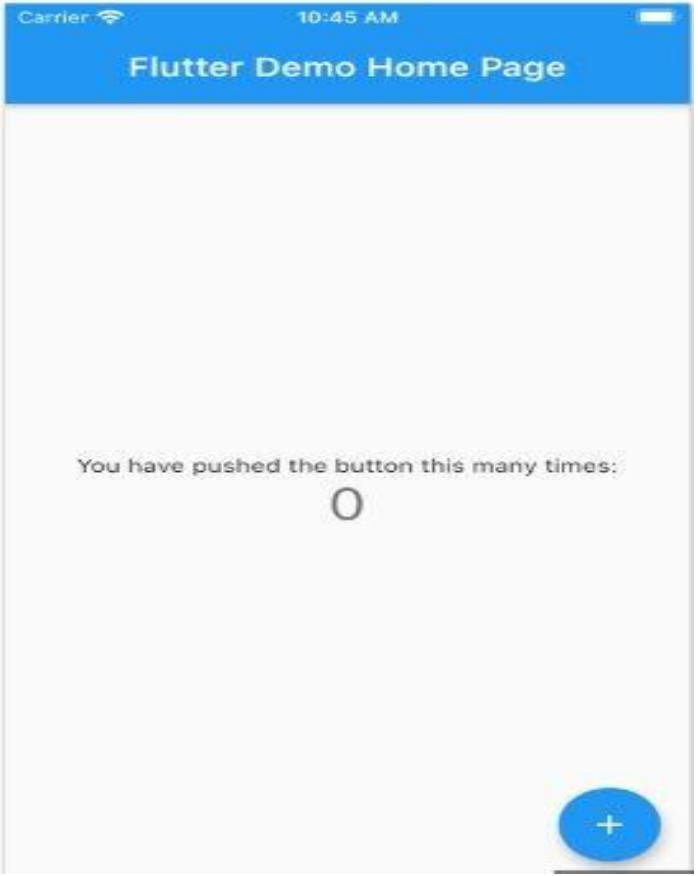
Each element on a screen of the Flutter app is a widget. Each widget is a part of a user interface

Widget Tree : position widgets within each other to bu simple and complex layouts



Widget Tree


MyApp
MaterialApp
MyHomePage
Scaffold
AppBar
Center
FloatingAction Button
Text
Column
Icon
Text
Text


Carrier
10:45 AM
Flutter Demo Home Page
You have pushed the button this many times:
0
+

Center

A widget that centers its child within itself.

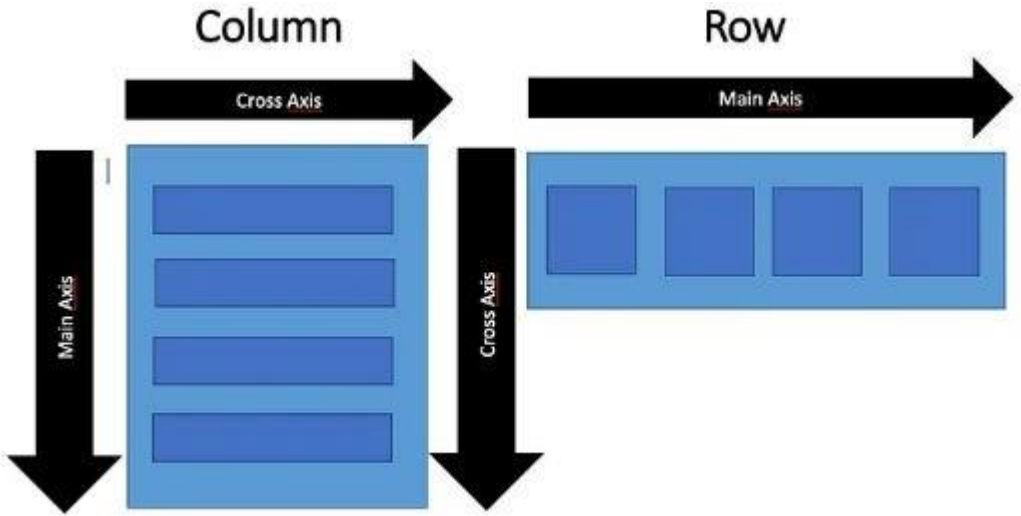
This widget will be as big as possible if its dimensions are constrained and widthFactor and heightFactor are null. If a dimension is unconstrained and the corresponding size factor is null then the widget will match its child's size in that dimension. If a size factor is non-null then the corresponding dimension of this widget will be the product of the child's dimension and the size factor. For example if widthFactor is 2.0 then the width of this widget will always be twice its child's width.

Column

A widget that displays its children in a vertical array.

To cause a child to expand to fill the available vertical space, wrap the child in an Expanded widget.

The Column widget does not scroll (and in general it is considered an error to have more children in a Column than will fit in the available room). If you have a line of widgets and want them to be able to scroll if there is insufficient room, consider using a ListView.



Category of Widgets:

There are mainly 14 categories in which the flutter widgets are divided. They are mainly segregated on the basis of the functionality they provide in a flutter application.

1. **Accessibility:** These are the set of widgets that make a flutter app more easily accessible.
2. **Animation and Motion:** These widgets add animation to other widgets.
3. **Assets, Images, and Icons:** These widgets take charge of assets such as display images and show icons.
4. **Async:** These provide async functionality in the flutter application.
5. **Basics:** These are the bundle of widgets that are absolutely necessary for the development of any flutter application.
6. **Cupertino:** These are the iOS designed widgets.
7. **Input:** This set of widgets provides input functionality in a flutter application.
8. **Interaction Models:** These widgets are here to manage touch events and route users to different views in the application.
9. **Layout:** This bundle of widgets helps in placing the other widgets on the screen as needed.
10. **Material Components:** This is a set of widgets that mainly follow material design by Google.
11. **Painting and effects:** This is the set of widgets that apply visual changes to their child widgets without changing their layout or shape.
12. **Scrolling:** This provides scrollability of to a set of other widgets that are not scrollable by default.
13. **Styling:** This deals with the theme, responsiveness, and sizing of the app.
14. **Text:** This displays text.

Types of Widgets:

fi. **Stateless Widget**

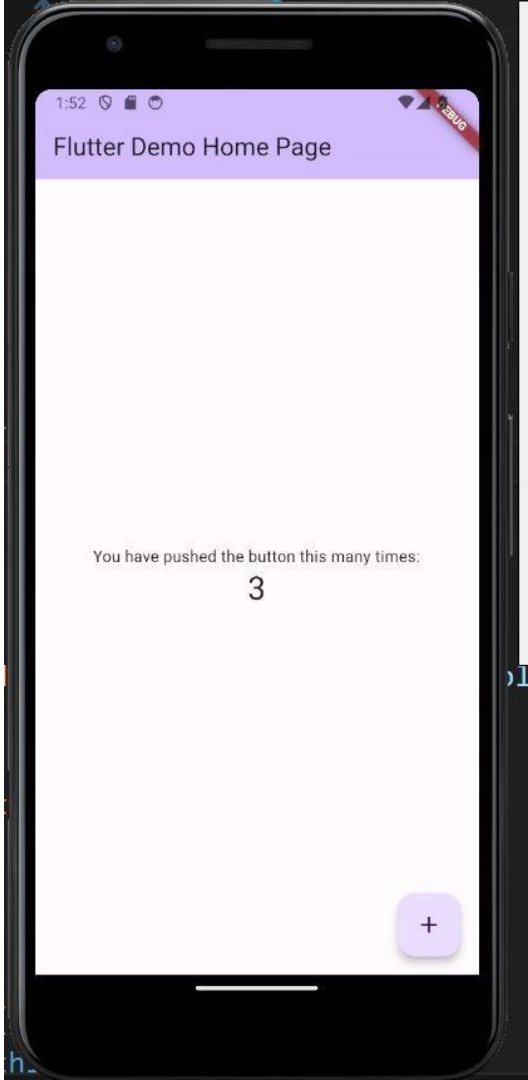
2. **Stateful Widget**

Stateful Widget	Stateless Widget
when Widget changes its value, that's Stateful. e.g. Checkbox, Radio button, Textfield	No change in widget value, that's Stateless. e.g. Text, Icon, Icon button, Raised button
Override the CreateState() and return State.	Override the build() and return Widget.
Use when user want to change UI dynamically.	Use when UI remains constant during runtime.
When Widget's state changes, the State object calls setState(), telling framework to redraw widget.	

Stateful Widget Vs Stateless Widget

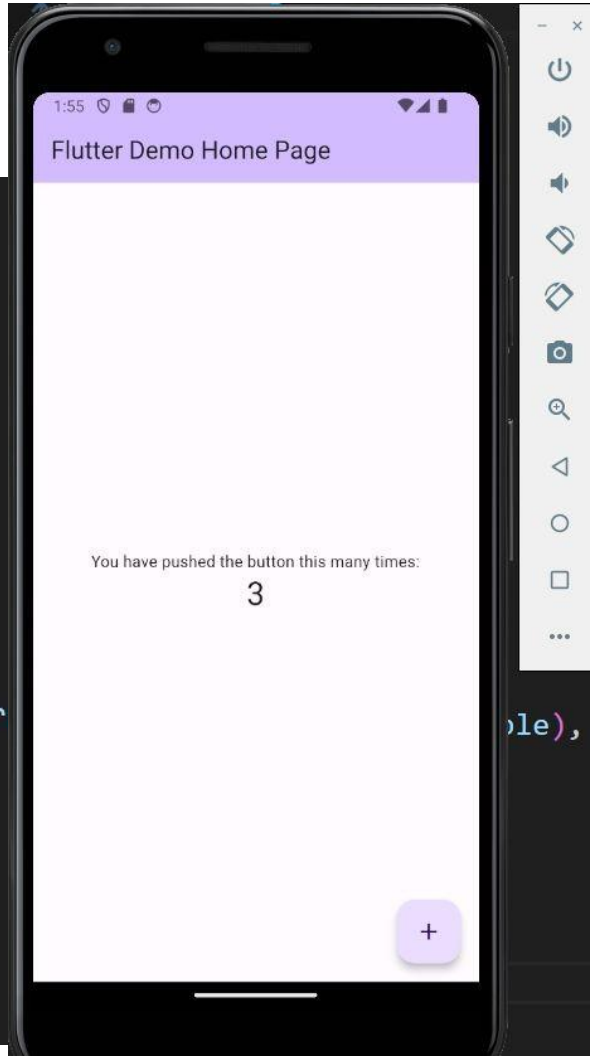
MaterialApp → widget that wraps a number of widgets

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ), // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
  }
}
```



Remove the debug Banner

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPur
        useMaterial3: true,
      ), // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
  }
}
```



Bool useMaterial3→ updated version than material 2 with more effects and colors

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ), // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
  }
}
```

`Home property:` displayed first when the application is started normally

StatefulWidget related to State Class

```dart
class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {

      _counter++;
    });
  }
```

fat arrow notation (=>). A fat arrow is used to define a single expression in a function. This is a cleaner way to write functions with a single statement.

_ means private member

Each time setState is called, the build function is re-excuted

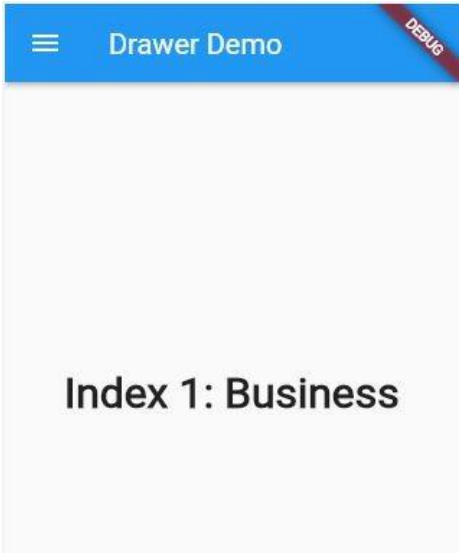
```dart
  @override
  Widget build(BuildContext context) {
    return Scaffold(
    appBar: AppBar(

      backgroundColor: Theme.of(context).colorScheme.inversePrimary,

      title: Text(widget.title),
    ), // AppBar
    body: Center(

      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        const Text(
          'You have pushed the button this many times:',
        ), // Text
        Text(
          '$_counter',
          style: Theme.of(context).textTheme.headlineMedium,
        ), // Text
      ], // <Widget>[]
      ), // Column
    ), // Center
    floatingActionButton: FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: const Icon(Icons.add),
    ), // This trailing comma makes auto-formatting nicer for build methods.
  ); // Scaffold
  }
}
```

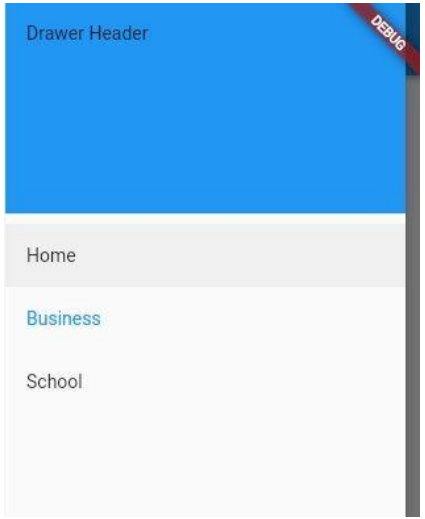
Scaffold

Scaffold is a class in flutter which provides many widgets

APIs like Drawer, Snack-Bar, Bottom-Navigation-Bar, Floating-Action-Button, App-Bar,

Scaffold will expand or occupy the whole device screen. It will occupy the available space.





Snackbar

_____ Explore the following link

https://docs.flutter.dev/cookbook/design/snackbars

Solve Exercise Snackbars on Moodle:

- Open New flutter project,
- Copy the snackbar code in an empty project
- Run the code
- Change the duration of the snackbar appearance
- Change the displayed message
- Remove the debug mode sign
- Upload the
- Upload the screenshot of the output
- Be ready for discussion with dr Taraggy

Drawer

____ Explore the following link

https://docs.flutter.dev/cookbook/design/drawer

Note the following:

- List
- TextStyle
- IconButton
- ListView
- ListTile
- Navigator.pop

Solve Exercise Drawer on Moodle

- Open new empty flutter project
- Copy the sample code from the link https://docs.flutter.dev/cookbook/design/drawer
- Add items to the list view
- Change the Colors and Font
- Upload the screenshot of your modified code
- Be ready for discussion with dr Taraggy

Simple Exercise (Fill in the blanks)

_____

```
class MyStatelessWidget extends StatelessWidget {
  @override
  Widget _________(BuildContext context) {
    return Text('I am a Stateless Widget');
  }
}
```

Simple Exercise (Fill in the blanks)

```
class MyStatefulWidget extends _____ {
  @override
_____createState() => _MyStatefulWidgetState();
}

class _____ extends State<MyStatefulWidget> {
  int counter = 0;

  @override
  Widget _____(BuildContext context) {
    return Column(
      children: [
        Text('Counter: $counter'),
        ElevatedButton(
          onPressed: () {
            _____(() {
              counter++;
            });
          },
          child: Text('Increment'),
        ),
      ],
    );
  }
}
```

_____