Day 1 Introduction To Flutter

MARLON I. TAYAG

Flutter

- Flutter is not a language (like JavaScript, for example). Flutter uses Dart for its language.
- Flutter is Google's mobile SDK / UI framework that enables developers to build native apps that run on Android and iOS devices. Developers write code in a single codebase that works on both platforms.

Benefits of Using Flutter Apps Dev

High Productivity

- Flutter was written for high productivity, to get apps out fast.
- You can change your code and hot reload the changes,without any kind of delay.
- Flutter includes the UI Widgets you need.
- Flutter works with most IDEs.

Benefits of Using Flutter Apps Dev

High Quality

The included Flutter UI Widgets work seamlessly and conventionally with the target platform. Scrolling, navigation, icons and fonts match the target system.

- When you write an Android app with the Flutter
- Widgets it looks like a normal Android app.
- When you write an iOS app with the Flutter Widgets, it looks like a normal iOS app..

Benefits of Using Flutter Apps Dev

High Performance

- The code you write in Flutter runs natively so it flies!

It is Free and Open

- Flutter is free and Open Source.

Fuschsia

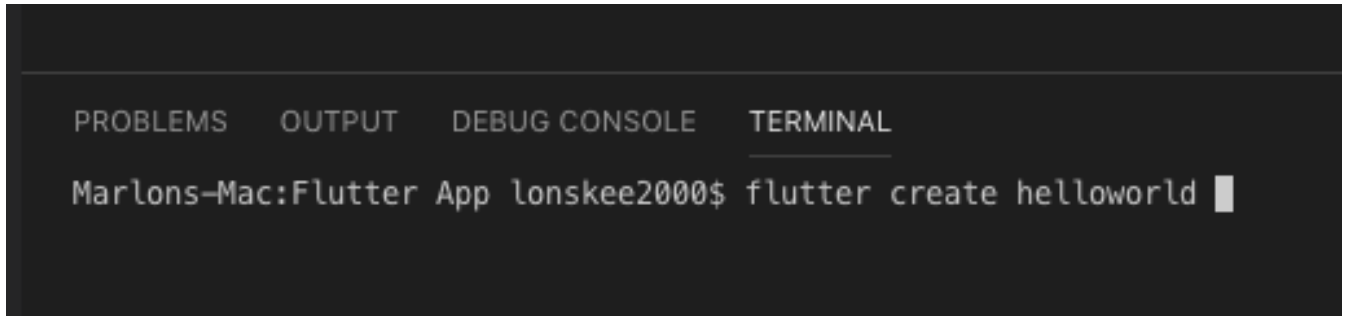
Fuschsia is Google's next Operating System for mobile devices. All of the apps for Fuschsia are being developed by Google in Flutter.

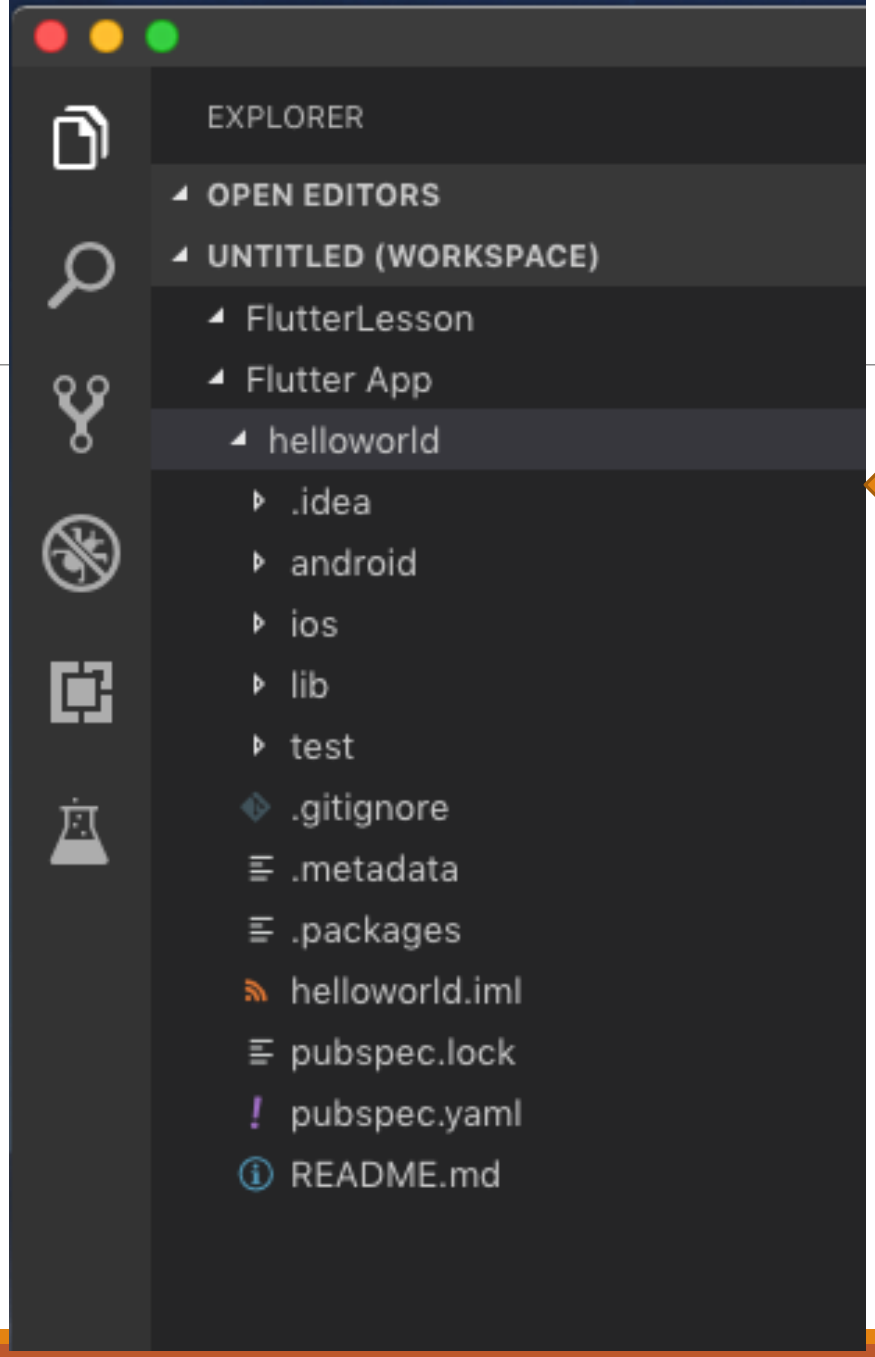
Software For App Dev

1. VS Studio
2. Flutter SDK
3. Dart Platform
4. Xcode Runtime
5. Android Emulator and iOS Emulator

Creating First App

1. Open VS Studio
2. Create a Folder on Documents -> Flutter App
3. Drag Flutter App inside VS Studio to Add Folder
4. Right-clicked Flutter App folder inside VS Studio and select **`Open in Terminal`**
5. Inside the terminal type **`flutter create helloworld`**





helloworld app with default application folders

Folders

The default Flutter application is organized into several folders

Folder	Description
[root]	Root folder. This usually contains configuration files. The most important of these configuration files is the 'pubspec.yaml' file, which declares the project dependencies.
.idea	Intellij project folder. Feel free to remove this folder if you are using Visual Studio Code.
android	As the name suggests, the folder contains all the Android-related files and code(s) for the application. This is where Android-specific settings and code resides. When building for Android, Flutter uses Gradle as the dependency manager.

Folders

The default Flutter application is organized into several folders

Folder	Description
build	This folder is created and used by gradle when you build the project.
ios	Similar to the ‘android’ folder, this folder contains the iOS related files and code(s) for the application.
lib	This is where the application code resides. You should see a file ‘**main.dart**’, the entry point for the Flutter application. This is the file you select and run. You will add more files and subfolders into this folder.

Folders

The default Flutter application is organized into several folders

Folder	Description
test	This is where the unit testing code resides. You may add more files and subfolders into this folder.

Emulators

These are great for developers, enabling them to develop their code to run on multiple devices, see how they look on each device. Later on, you can use the real hardware for final pre-release testing.

Open Emulator / Simulator

iOS Emulator

```
Open -a Simulator.app
```

Android Emulator

```
emulator -avd {AndroidVM}
```

Running the App

To run the flutter app type

```
flutter run
```

Note: You must be inside the project folder of your app

Hot Restarting & Reloading

Hot Restarting

This loads your changed code into the Dart VM and restarts the application. This is the safest thing to do and doesn't take long.

Hot Reloading

If you want to load your changed code into the Dart VM but you don't want to change the application state, you can do this. The result might be different behavior vs a hot restart.

If you are using 'flutter' run to run the app from the command line, you can use the key '**R**' to hot restart and the key '**r**' to hot reload.

Dependencies & Packages

The Dart ecosystem uses *packages* to manage shared software such as libraries and tools. To get Dart packages, you use the **pub package manager**. You can find publicly available packages on the **Pub site**, or you can load packages from the local file system or elsewhere, such as Git repositories. Wherever your packages come from, pub manages version dependencies, helping you get package versions that work with each other and with your SDK version.

Non-Core Packages

Flutter comes with many packages by default. These are called Core Packages and you don’t need to declare any kind of external dependency to use them

Flutter Boilerplate Code

main.dart

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
  }
}

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

  // This widget is the home page of your application. It is stateful, meaning
  // that it has a State object (defined below) that contains fields that affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in this
  // case the title) provided by the parent (in this case the App widget) and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".

  final String title;

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

class _MyHomePageState extends State<MyHomePage> {
```

How do we make Flutter apps?

We build widgets that control UI elements on the screen

We mix and match widgets to build the desired UI for the app we're making

Some widgets are provided by Flutter

Some are created by you and me

Widgets

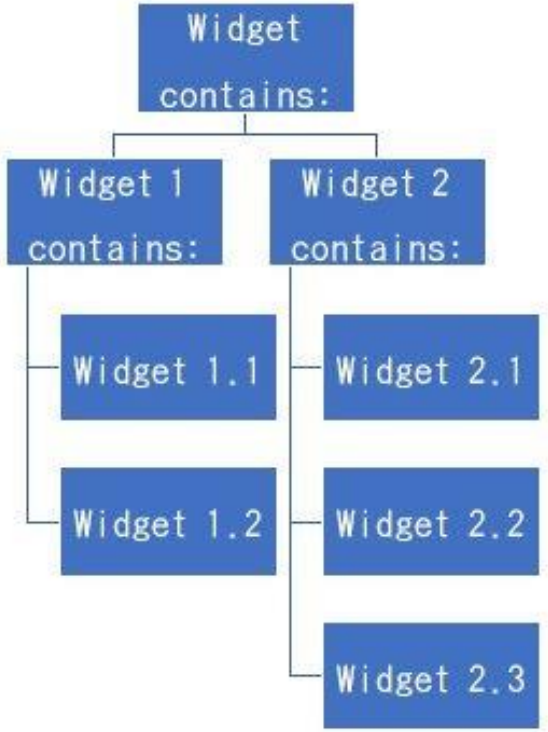
Widgets are the Building Blocks of your UI. Whenever we build a user interface in Flutter, it is composed of Widgets. Putting your widgets together is called Composition. Think of a user interface as a jigsaw. Each widget is a piece of the puzzle:

```
class PaddedText extends StatelessWidget {

 final String _data;

 PaddedText(this._data, {Key key})
      : super(key: key);

 @override
 Widget build(BuildContext context) {
   return new Padding(
       padding: const EdgeInsets.all(4.0),
       child: new Text(_data)
   );
 }
}
```

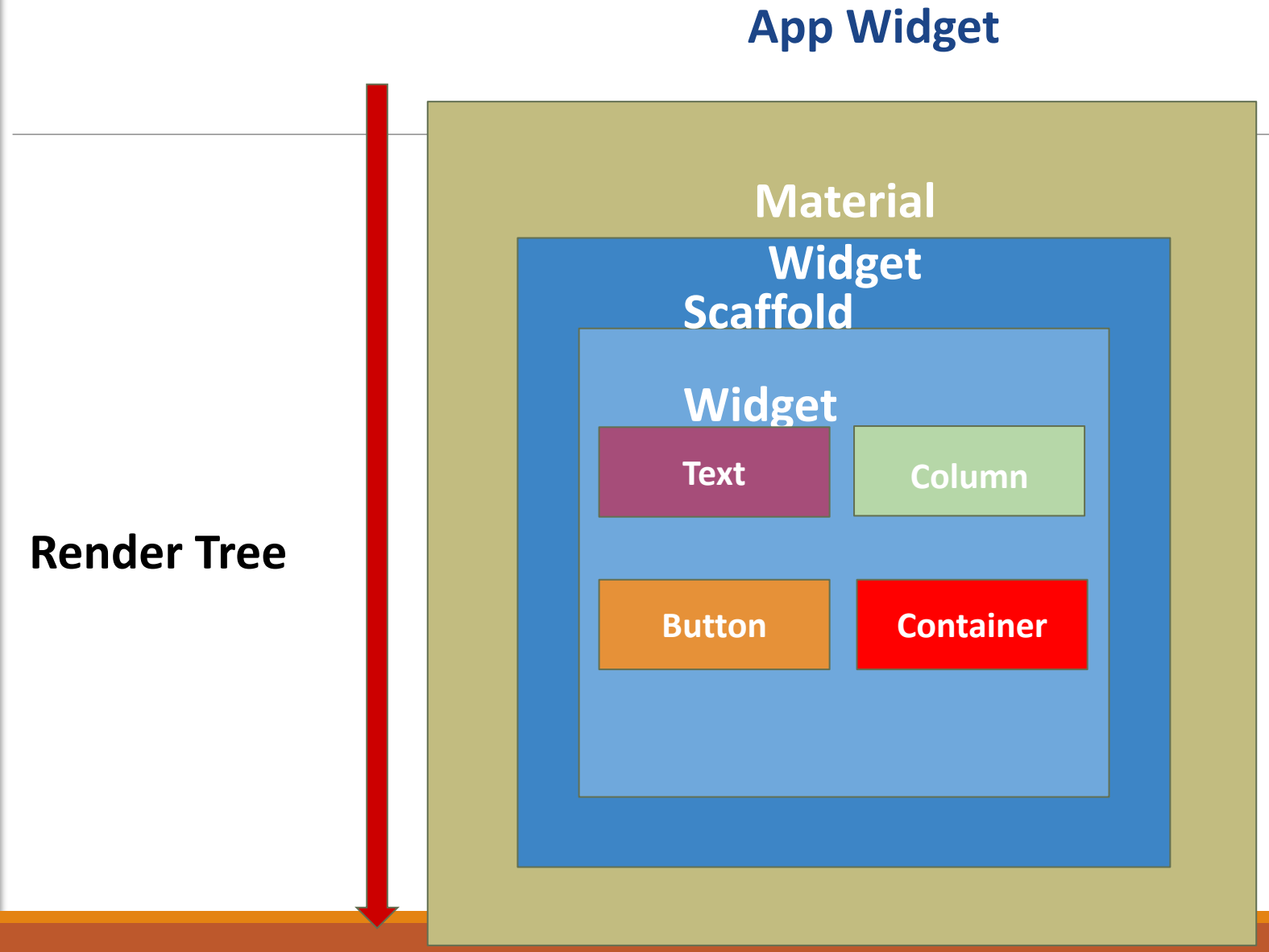
Everything is a Widget

Widget Tree

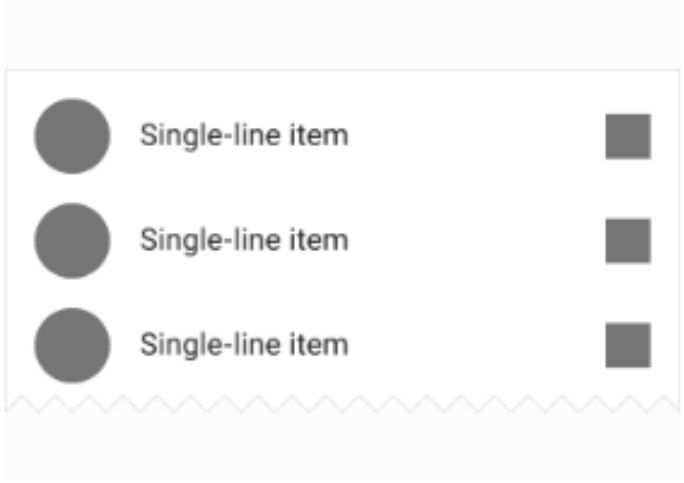
Unlike a Jigsaw, a widget can contain other widgets, in a tree structure, a hierarchy. This is often called a Widget Tree.



Everything is a Widget



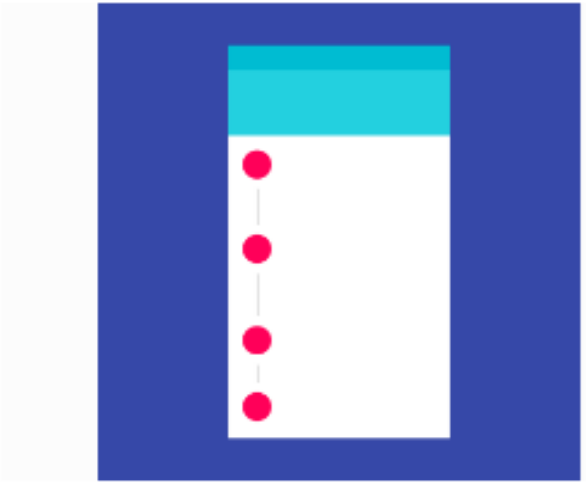
Layout



ListTile

A single fixed-height row that typically contains some text as well as a leading or trailing icon.

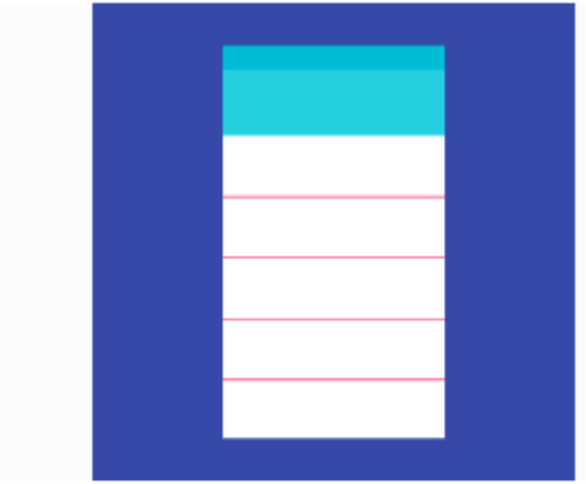
Documentation



Stepper

A material stepper widget that displays progress through a sequence of steps.

Documentation



Divider

A one logical pixel thick horizontal line, with padding on either side.

Documentation

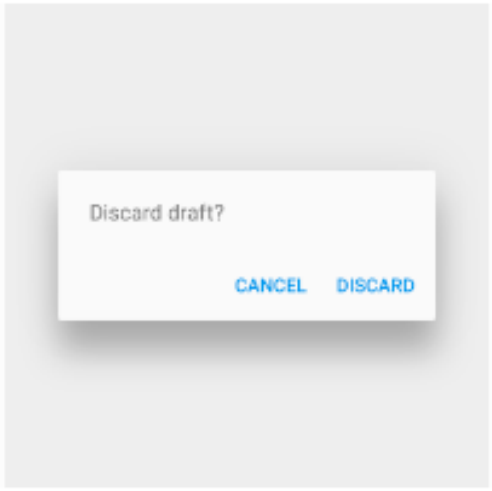
https://flutter.io/widgets/widgetindex/

Dialogs, alerts, and panels

SimpleDialog

Simple dialogs can provide additional details or actions about a list item. For example they can display avatars icons clarifying subtext or orthogonal actions...

Documentation



AlertDialog

Alerts are urgent interruptions requiring acknowledgement that inform the user about a situation. The AlertDialog widget implements this component.

Documentation

BottomSheet

Bottom sheets slide up from the bottom of the screen to reveal more content. You can call showBottomSheet() to implement a persistent bottom sheet or...

Documentation

https://flutter.io/widgets/widgetindex/

Basics

Widgets you absolutely need to know before building your first Flutter app.

VISIT

Material Design

Visual, behavioral, and motion-rich widgets implementing Google's Material Design guidelines.

VISIT

Cupertino (iOS-style widgets)

Beautiful and high-fidelity widgets for current iOS design language.

VISIT

Layout

Arrange other widgets columns, rows, grids, and many other layouts.

VISIT

Text

Display and style text.

VISIT

Assets, Images, and Icons

Manage assets, display images, and show icons.

VISIT

Input

Take user input in addition to input widgets in in Material Design and Cupertino.

VISIT

Animation and Motion

Bring animations to your app. Check out Animations in Flutter for an overview.

VISIT

Interaction Models

Respond to touch events and route users to different views.

VISIT

Styling

Manage the theme of your app, makes your app responsive to screen sizes, or add padding.

VISIT

Painting and effects

These widgets apply visual effects to the children without changing their layout, size, or position.

VISIT

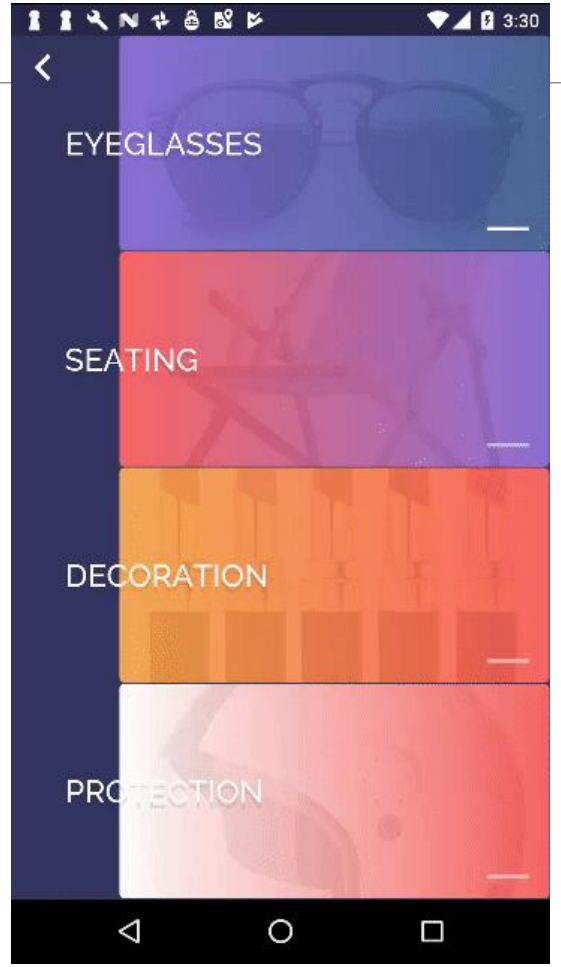
Async

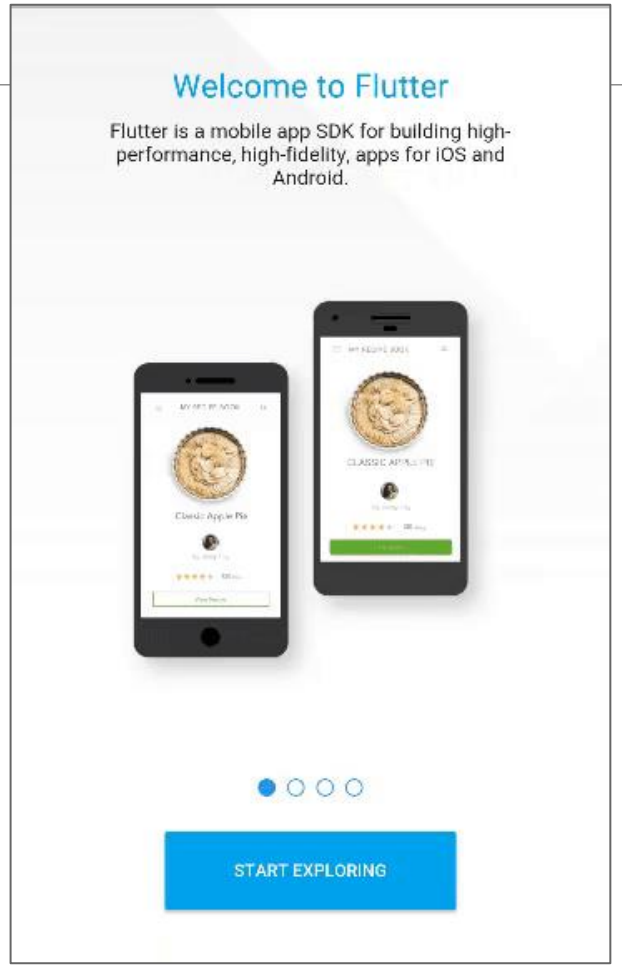
Async patterns to your Flutter application.

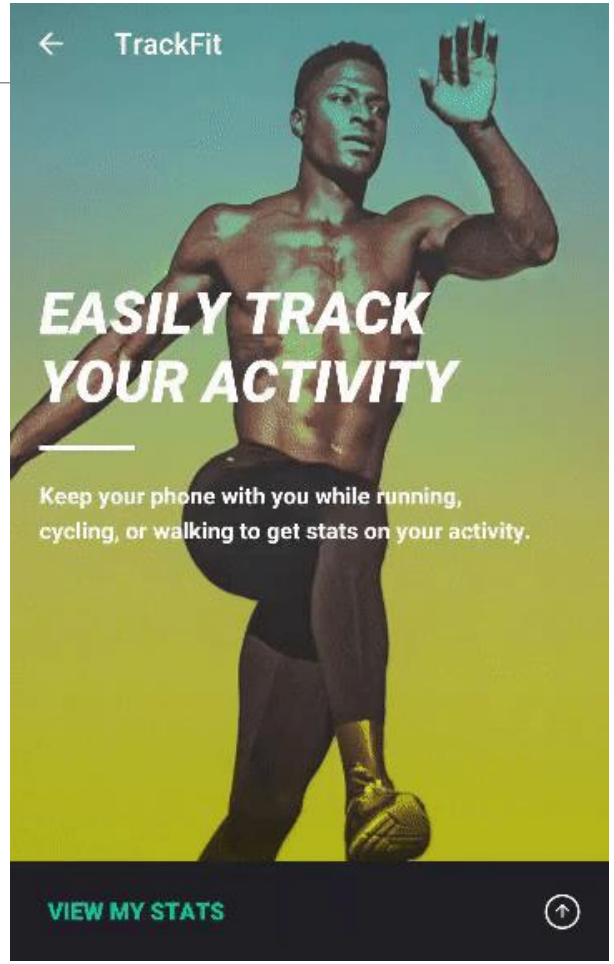
VISIT

https://flutter.io/widgets/

Great looking and fast Widgets







Creating the real HELLOWORLD app

Coding Steps helloword app

1. Delete all template code
2. Import library (**widget.dart**)
3. Create **main()**
4. Implement widgets
   a. Center
   b. Text
   c. TextStyle
5. Use runApp() to load app

main.dart ×

helloworld ▸ lib ▸ main.dart ▸ ...

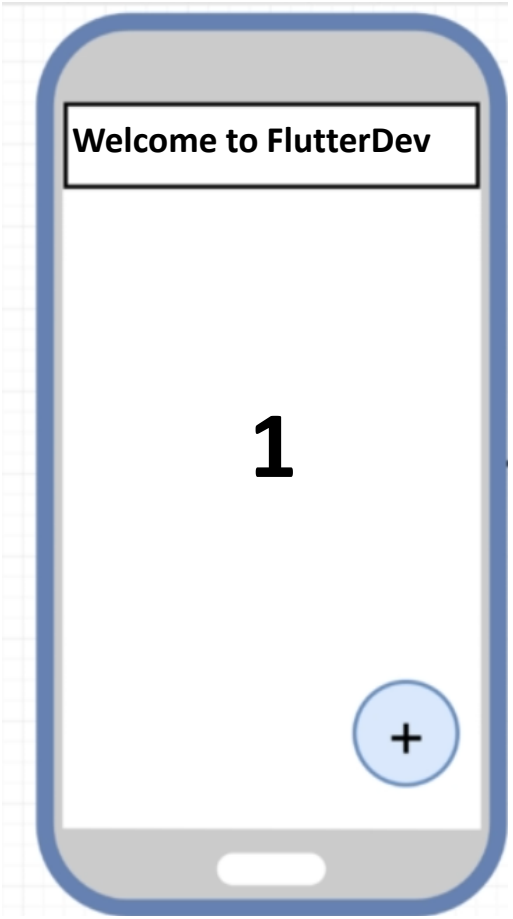
```dart
import 'package:flutter/widgets.dart';

void main(){

    runApp(

        Center(
          child: Text('Hello World',textDirection: TextDirection.ltr,) ,

        ) // Center
    );


}

```

Using TextStyle (helloworld)

helloworld ▸ lib ▸ main.dart ▸ main

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

void main(){

    runApp(

        Center(
          child: Text('Hello World',textDirection: TextDirection.ltr,
          style: TextStyle(
                    fontSize: 50.00,fontWeight: FontWeight.bold,
          ), // TextStyle
          ) , // Text
        ) // Center
    );
}
```

Target App



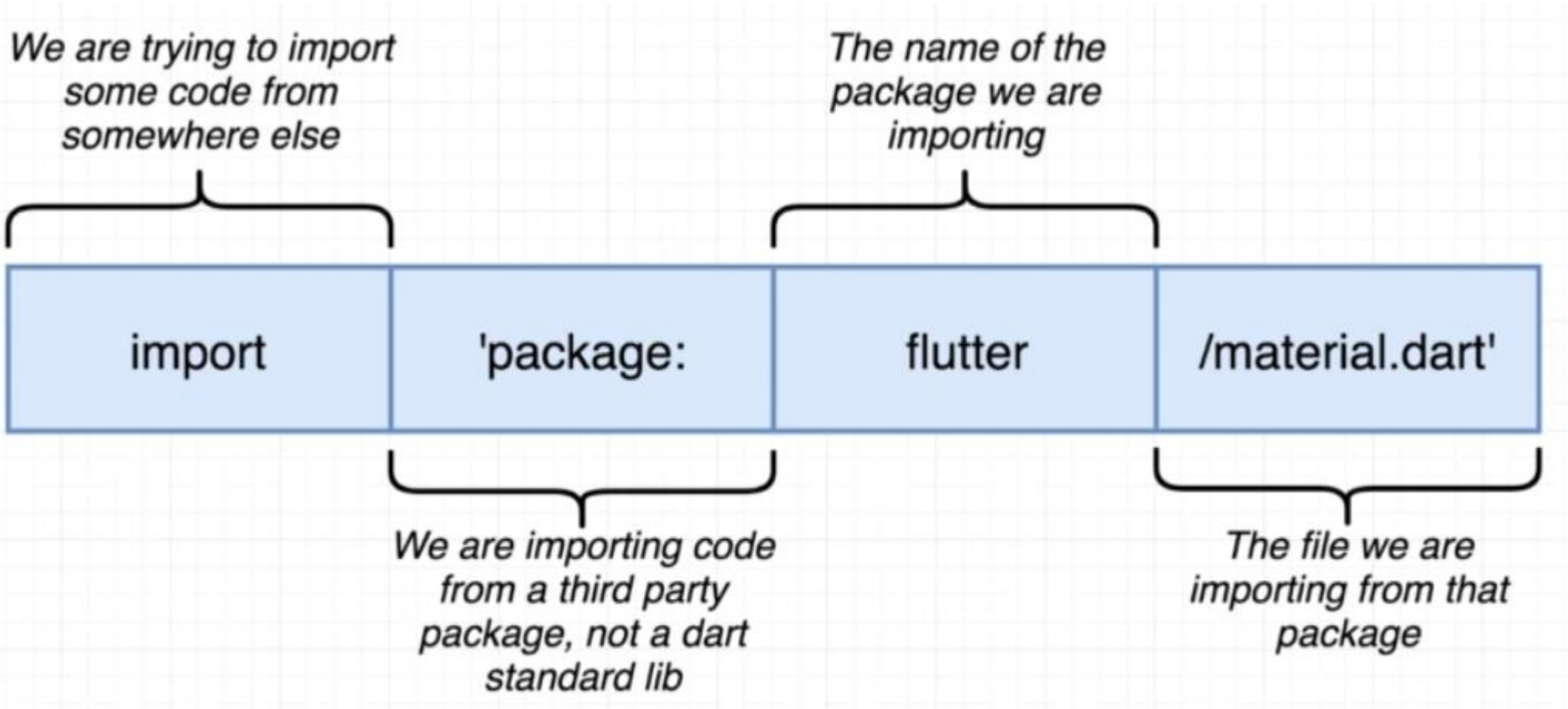
Four step design process

1. Import helper library from flutter to get content on the screen
2. Define a ‘main’ function to run when our apps starts
3. Create a new text widget to show some text on the screen
4. Take the widget and get it on the screen

Import Statements

```
1    //I need to import a helper library
2    //from flutter to get content on the screen
3    import 'package:flutter/material.dart';
4
5    //Define a 'main' function to run when our app starts
6
7
8    //Create a text widget to show some text on the screen
9
10
11   //Take that widget and get it on the screenS
```

Import Statements



To remove Debug Banner on App

```
var app = MaterialApp(
    title: 'Flutter App',
    debugShowCheckedModeBanner: false,
    theme: ThemeData(
      primaryColor: Colors.green,
      accentColor: Colors.orange
    ), // ThemeData
```

Add debugShowCheckedModeBanner: false,

Creating Main Function

```dart
//I need to import a helper library
//from flutter to get content on the screen
import 'package:flutter/material.dart';

//Define a 'main' function to run when our app starts

void main () {
//Create a text widget to show some text on the screen


//Take that widget and get it on the screenS


}

```

Creating A Widgets

```dart
//I need to import a helper library
//from flutter to get content on the screen
import 'package:flutter/material.dart';

//Define a 'main' function to run when our app starts

void main () {
//Create a text widget to show some text on the screen
var app =MaterialApp(
  home: Text('Hi there!'),
); // MaterialApp

//Take that widget and get it on the screenS
```

Show the Widget on the Screen

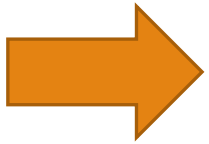
```dart
//I need to import a helper library
//from flutter to get content on the screen
import 'package:flutter/material.dart';

//Define a 'main' function to run when our app starts

void main () {
//Create a text widget to show some text on the screen
var app =MaterialApp(
  home: Text('Hi there!'),
); // MaterialApp

//Take that widget and get it on the screenS
runApp(app);
```

Run the App

1. Go to terminal type flutter run
2. To hot reload press Shift+R

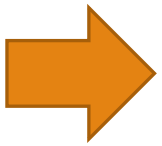
Using Scaffold Class

```dart
//I need to import a helper library
//from flutter to get content on the screen
import 'package:flutter/material.dart';

//Define a 'main' function to run when our app starts

void main() {
//Create a text widget to show some text on the screen
  var app = MaterialApp(
    home: Scaffold(
      appBar: AppBar(),
    ), // Scaffold
  ); // MaterialApp
  runApp(app);
//Take that widget and get it on the screen
}
```



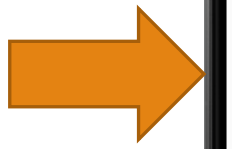
Customizing AppBar

```
1   //I need to import a helper library
2   //from flutter to get content on the screen
3   import 'package:flutter/material.dart';
4
5   //Define a 'main' function to run when our app starts
6
7   void main() {
8   //Create a text widget to show some text on the screen
9     var app = MaterialApp(
10      home: Scaffold(
11        appBar: AppBar(
12          title: Text("Let's see some images"),
13        ), // AppBar
14      ), // Scaffold
15    ); // MaterialApp
16    runApp(app);
17  //Take that widget and get it on the screen
18  }
19
```



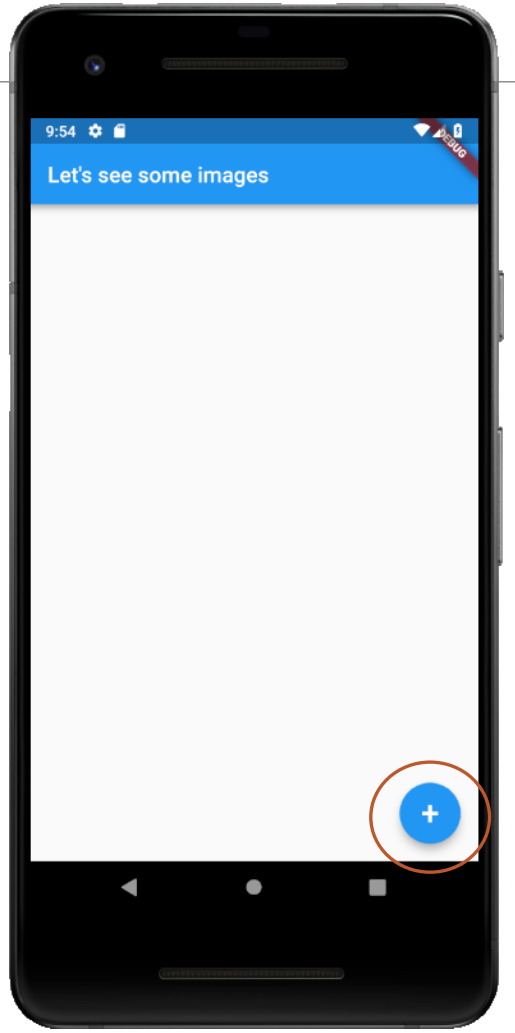
Adding FloatingActionButton (Text)

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

//Define a 'main' function to run when our app starts

void main() {
//Create a text widget to show some text on the screen
  var app = MaterialApp(
    home: Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          print('Hi there');
        },

        child: Text('+'),


      ), // FloatingActionButton
      appBar: AppBar(
        title: Text("Let's see some images"),

      ), // AppBar
    ), // Scaffold
  ); // MaterialApp
  runApp(app);
//Take that widget and get it on the screen
}
```



Adding FloatingActionButton (TextStyle)

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

//Define a 'main' function to run when our app starts

void main() {
//Create a text widget to show some text on the screen
  var app = MaterialApp(
    home: Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          print('Hi there');
        },

        child: Text('+',style: TextStyle(fontSize: 30.0),),


      ), // FloatingActionButton
      appBar: AppBar(
        title: Text("Let's see some images"),

      ), // AppBar
    ), // Scaffold
  ); // MaterialApp
  runApp(app);
//Take that widget and get it on the screen
}
```



Adding FloatingActionButton (Icon)

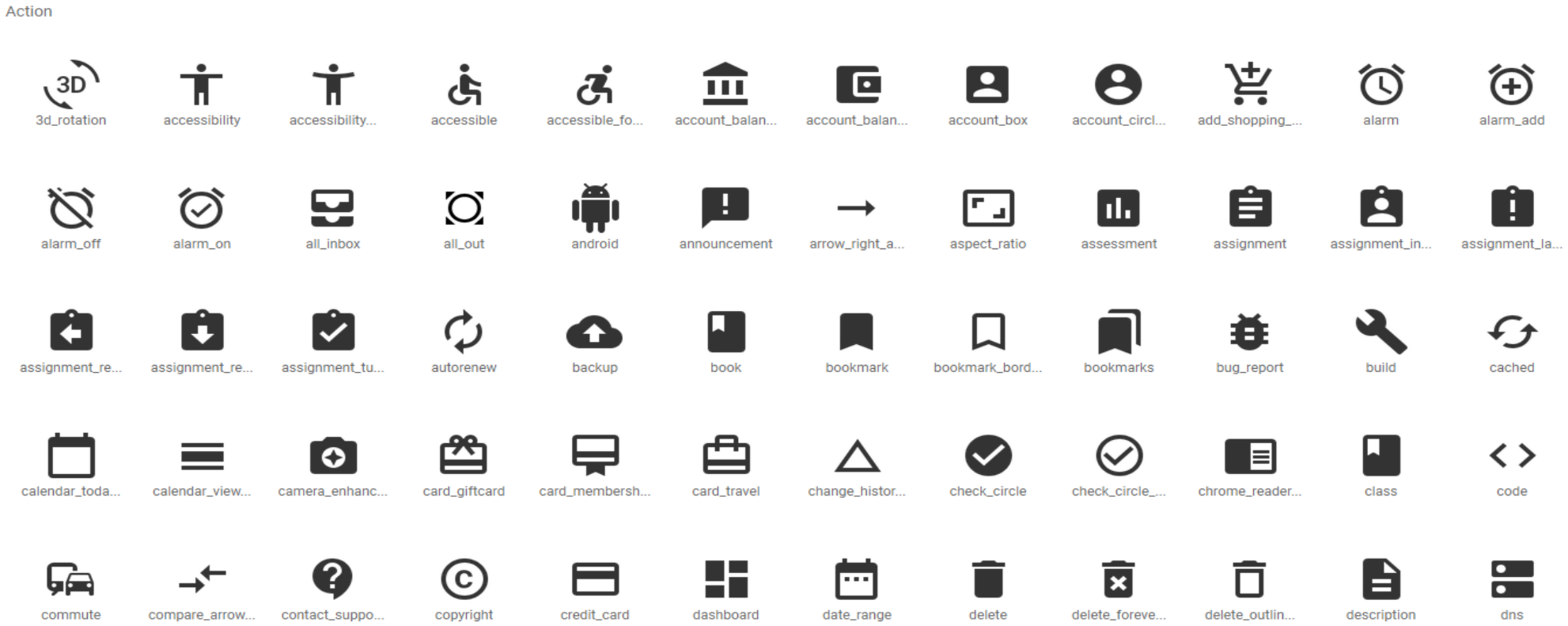
```dart
import 'package:flutter/material.dart';

//Define a 'main' function to run when our app starts

void main() {
//Create a text widget to show some text on the screen
  var app = MaterialApp(
    home: Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          print('Hi there');
        },

        child: Icon(Icons.add)


      ), // FloatingActionButton
      appBar: AppBar(
        title: Text("Let's see some images"),

      ), // AppBar
    ), // Scaffold
  ); // MaterialApp
  runApp(app);
//Take that widget and get it on the screen
}
```

Activity: Change Icon Property

1. Change the icon background to black
2. Change the icon color to white

Icon Material Design

https://material.io/tools/icons

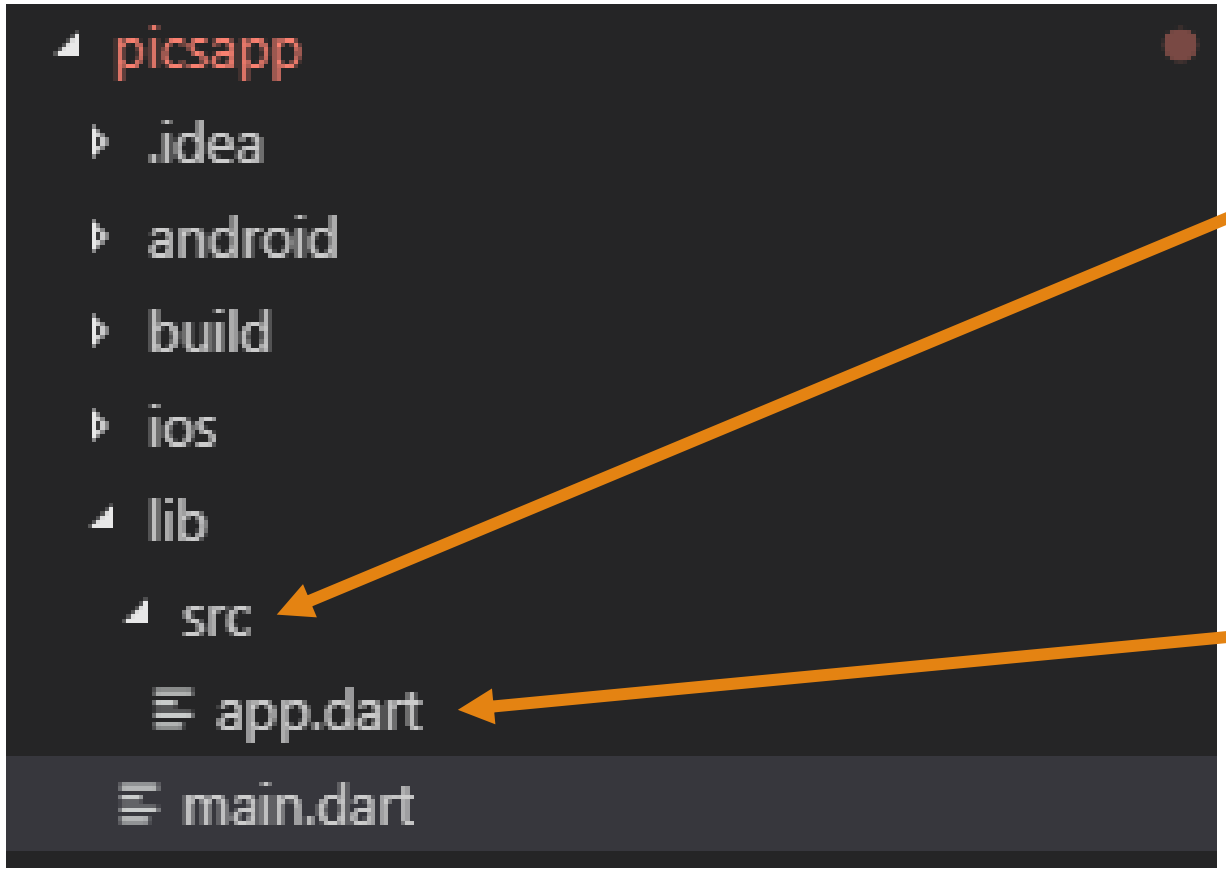


Creating Custom Widget and Working With Stateless Widget

StatelessWidget

This widget will have no instance variables that will change

Creating Custom Widget



1. Create a new folder **src**

2. Create a new file **app.dart**

Creating Custom Widget (app.dart)

```dart
//Import flutter helper library
import 'package:flutter/material.dart';


//Create a class that will be our custom widget
//This class must extend the 'StatelessWidget' base class

class App extends StatelessWidget{
  @override
  Widget build (BuildContext context){
    return MaterialApp(
    home: Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          print('Hi there');
        },

        child: Icon(Icons.add)


      ), // FloatingActionButton
      appBar: AppBar(
        title: Text("Let's see some images"),

      ), // AppBar
    ), // Scaffold
  );   // MaterialApp
  }
}
```

1. Import flutter library

2. Create a extended StatelessWidget

3. Copy code from main.dart

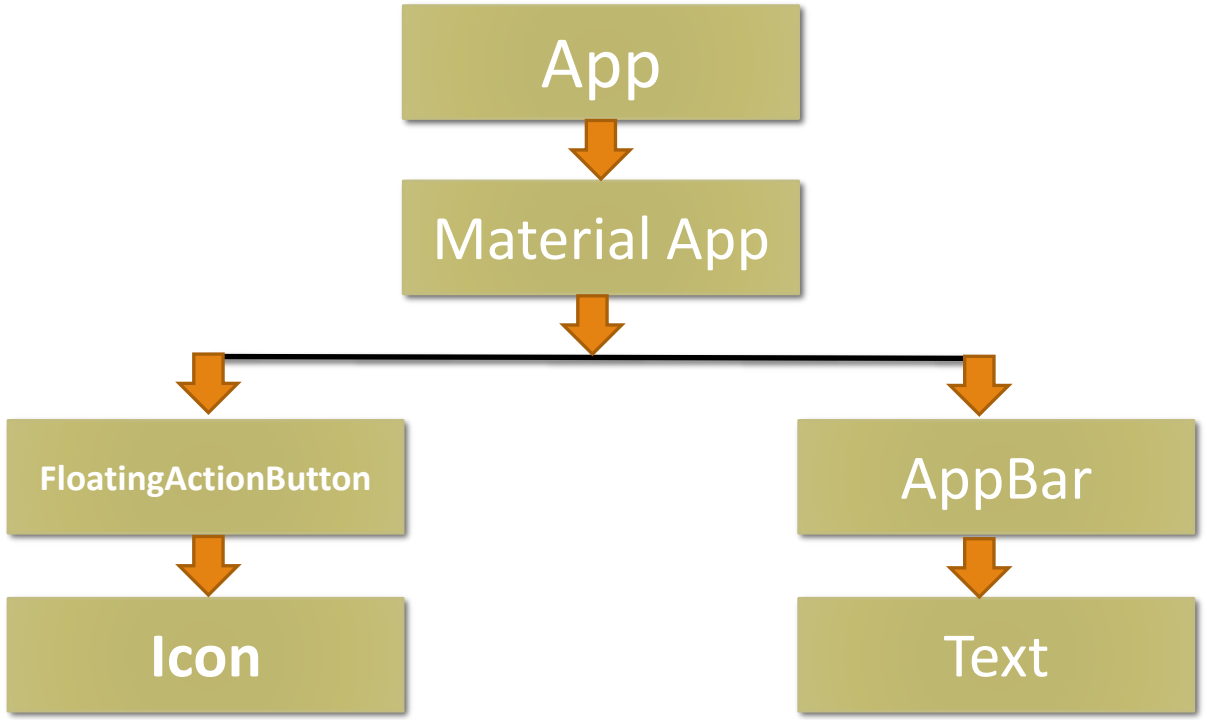
Creating Custom Widget (main.dart)

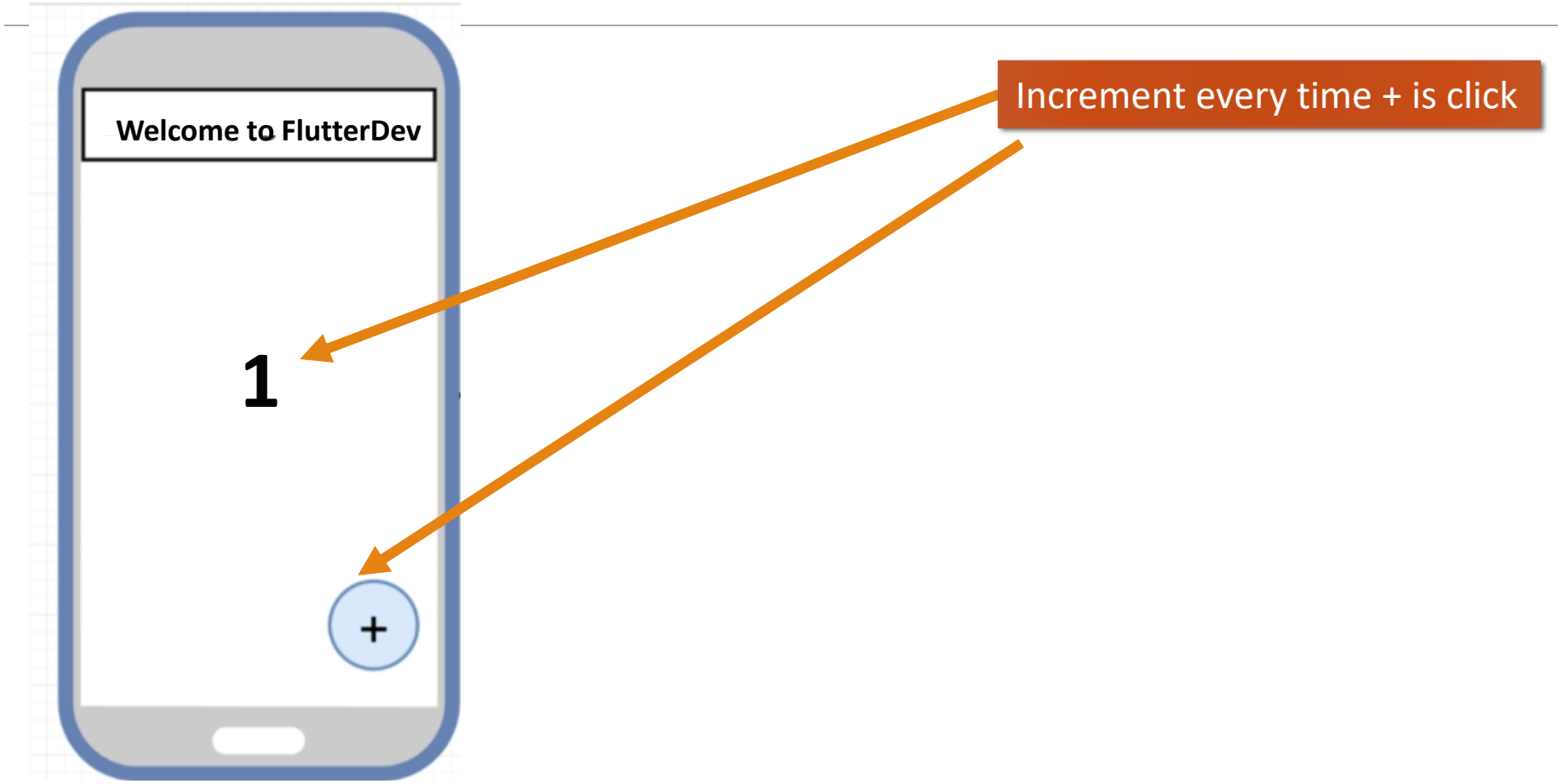
```dart
//I need to import a helper library
//from flutter to get content on the screen
import 'package:flutter/material.dart';
import 'src/app.dart';
//Define a 'main' function to run when our app starts

void main() {
//Create a text widget to show some text on the screen
 var app = new App();
 runApp(app);
//Take that widget and get it on the screen5
}
```

1. Import **app.dart** using relative PATH

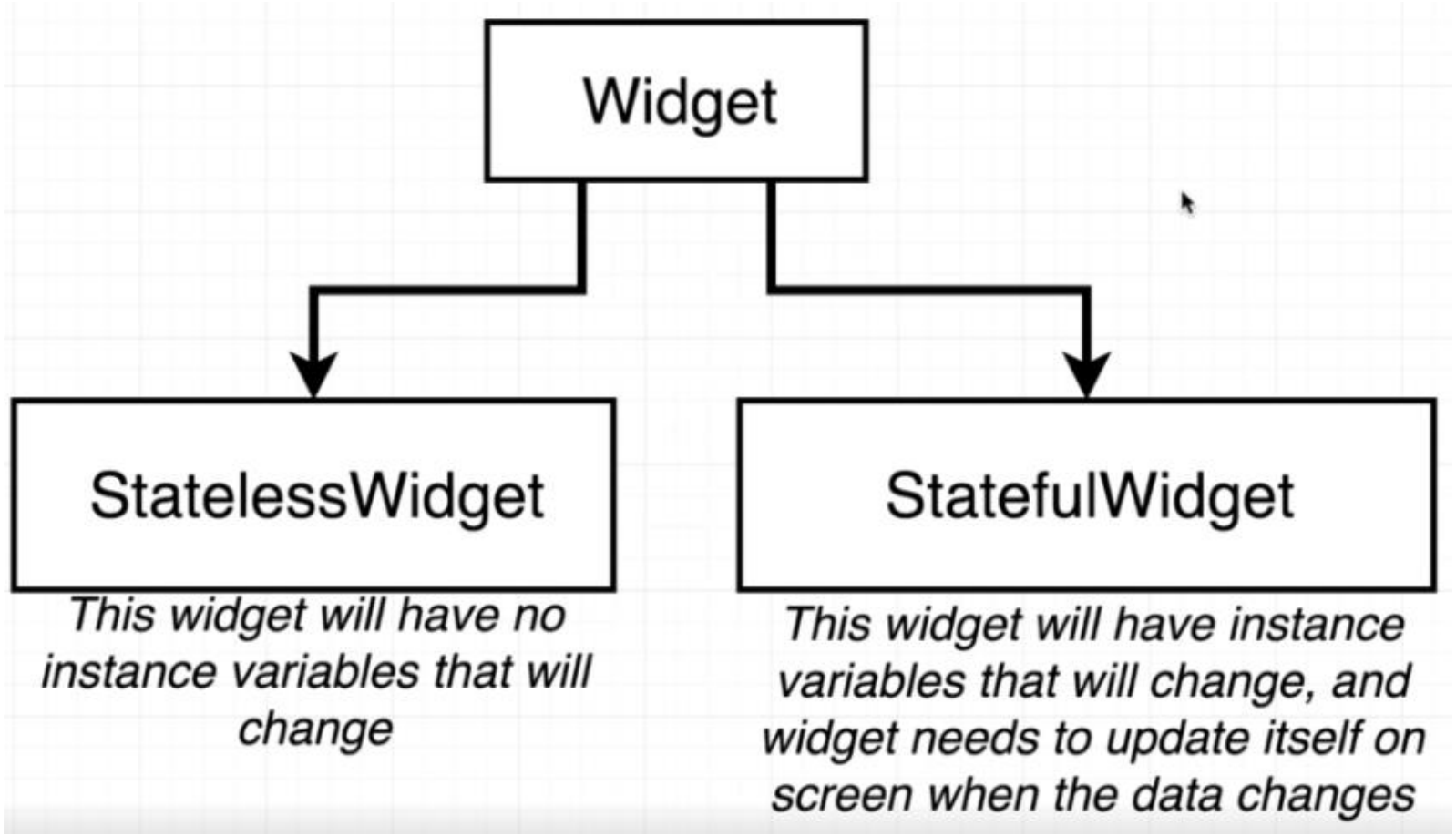
1. Declare and use class App()

Widget Tree

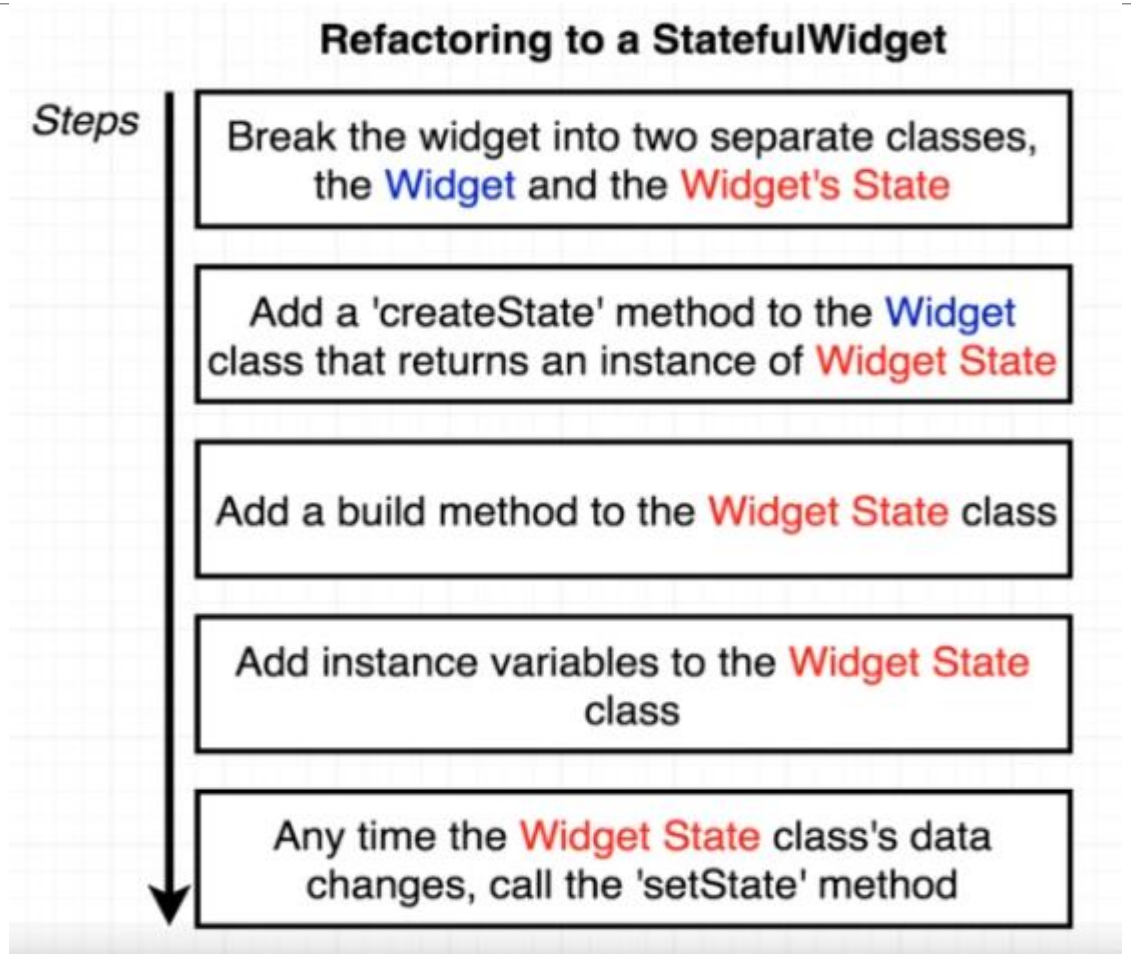



Welcome to FlutterDev
1
+
Increment every time + is click

Refactor StatelessWidget to StatefulWidget



Refactor StatelessWidget to StatefulWidget



Refactor StatelessWidget to StatefulWidget

```
class App extends StatefulWidget {



}

class AppState extends State<App>{
```

Break the widget into two separate classes, the Widget and the Widget's State

Refactor StatelessWidget to StatefulWidget

```
class App extends StatefulWidget {
  createState(){
    return AppState();
  }
}

class AppState extends State<App>{
```

Add a 'createState' method to the Widget class that returns an instance of Widget State

Refactor StatelessWidget to StatefulWidget

```
Widget build (BuildContext context){
  return MaterialApp(
  home: Scaffold(
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        print('Hi there');
      },

      child: Icon(Icons.add)


    ), // FloatingActionButton
    appBar: AppBar(
      title: Text("Let's see some images"),

    ), // AppBar
  ), // Scaffold
);   // MaterialApp
```

Add a build method to the Widget State class

Refactor StatelessWidget to StatefulWidget

```
class AppState extends State<App>{
  @override

  int counter = 0;

  Widget build (BuildContext context){
    return MaterialApp(
    home: Scaffold(
```

Add instance variables to the Widget State class

Refactor StatelessWidget to StatefulWidget

```dart
class AppState extends State<App>{
  @override

  int counter = 0;

  Widget build (BuildContext context){
    return MaterialApp(
    home: Scaffold(
      body: Text('$counter'),
      floatingActionButton: FloatingActionButton(
        onPressed: () {

          setState(() {
            counter +=1;
          });
        },

        child: Icon(Icons.add)


      ), // FloatingActionButton
      appBar: AppBar(
        title: Text("Let's see some images"),

      ), // AppBar
    ), // Scaffold
  );    // MaterialApp
  }
}
```

Show the value on the screen

Any time the Widget State class's data changes, call the 'setState' method

Align Text Using Container

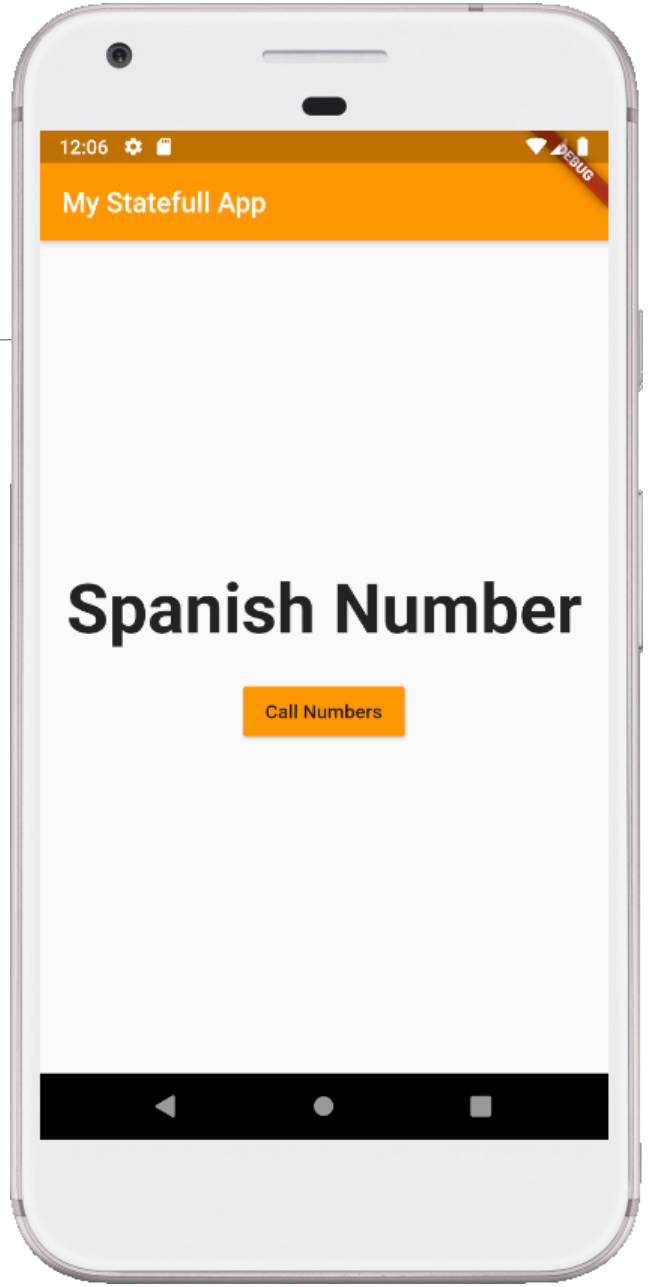
```
body:  Container(
  child: Align(alignment: Alignment.center,
  child: Text('$counter', style: TextStyle(color: Colors.blueAccent,fontSize: 40.0,fontWeight: FontWeight.bold),),), // Align
), // Container
```

Activity: Change Color (Blue and Red)

Change the color of the item RED the value of the item is greater than 10

Spanish Number App (Working With Stateful Widget)

1. Create an app that randomly show a number in Spanish
2. Widget to used:
   a. Stateful Widget
   b. MaterialApp
   c. RaiseButton
   d. Scaffold
      i. AppBar
      ii. Text



1. Import Library and create Main()

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

void main() {
  runApp(MaterialApp(
      home: mySpanishNumber(),

    ) // MaterialApp
  );
}
```

2.Create the Stateful Widgets

```
10
11  class mySpanishNumber extends StatefulWidget {
12    mySpanishNumber({Key key}) : super(key: key);
13
14    mySpanishNumberState createState() => mySpanishNumberState();
15  }
```

mySpanishNumber

```
16
17   class mySpanishNumberState extends State<mySpanishNumber> {
18
19     int counter = 0;
20     int dispCount=0;
21     List<String> spanishNumbers = [
22         "uno",
23         "dos",
24         "tres",
25         "cuatro",
26         "cinco",
27         "seis",
28         "seite",
29         "otso",
30         "nueve",
31         "diez"
32     ];
```

mySpanishNumberState (1)

```
33
34  String defaultText = "Spanish Number";
35
36   void displayNumber (){
37     setState(() {
38       defaultText = spanishNumbers[counter];
39       if (counter < 9) {
40         counter = counter + 1;
41         dispCount++;
42       } else {
43         counter=0;
44         dispCount=0;
45       }
46
47     });
48   }
49
50   @override
51
52   Widget build(BuildContext context) {
53     return  Scaffold(
54       appBar: AppBar(
55         title: Text('My Statefull App'),
56         backgroundColor: Colors.orange,
57
58       ), // AppBar
59       body: Container(
60         child: Center(
61           child: Column(
62             mainAxisAlignment: MainAxisAlignment.center,
63             children: <Widget>[
64               Column(
65                 children: <Widget>[
66                   Text(defaultText,style: TextStyle(fontSize: 50.00, fontWeight: FontWeight.bold),),
```

mySpanishNumberState (2)

```
67
68
69                  ], // <Widget>[]
70                ), // Column
71                Padding(padding: EdgeInsets.all(10.00),),
72                RaisedButton(
73                    child: Text('Call Numbers'),
74
75                    //ADDING A SHAPE TO THE BUTTON
76                    /* shape: RoundedRectangleBorder(
77                        side: BorderSide(color: Colors.blueAccent),
78                        borderRadius: BorderRadius.circular(30),
79                    ),
80  */
81
82                    onPressed: displayNumber,
83
84                    color: Colors.orange,
85
86                ) // RaisedButton
87              ], // <Widget>[]
88            ), // Column
89          ), // Center
90        ), // Container
91
92      ); // Scaffold
93    }
94
95  }
```

mySpanishNumberState (3)

Spanish Number Code (main.dart)