Introduction To Flutter

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.

Software For App Dev

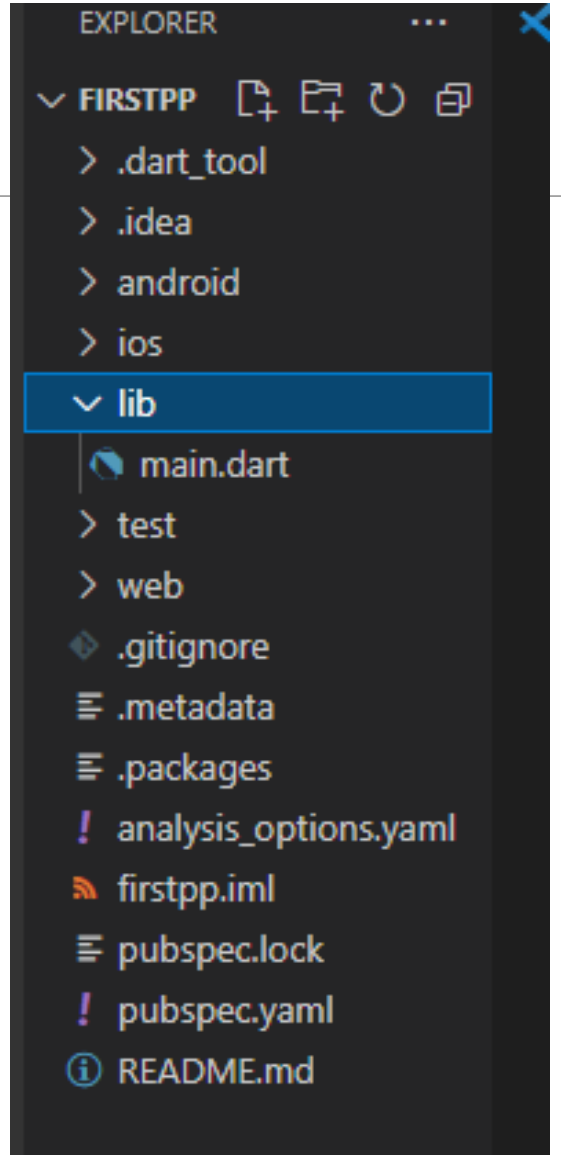
1. VS Studio
2. Flutter SDK
3. Dart Platform
4. Xcode Runtime
5. Android Emulator and iOS Emulator

Creating First App

1. Create a Folder Flutter Projects in drive c:
2. To start a project: Open terminal widow(cmd) and type in : flutter create <project name>.
   Example: flutter create firstapp
3. Change your current working directory to the project you just created. ***cd firstapp***
4. Type code<space> . To automatically load flutter app to vscode

```
c:\Flutter Projects>flutter create firstpp
Creating project firstpp...
  firstpp\lib\main.dart (created)
  firstpp\pubspec.yaml (created)
  firstpp\README.md (created)
  firstpp\test\widget_test.dart (created)
  firstpp\.gitignore (created)
  firstpp\.idea\libraries\Dart_SDK.xml (created)
  firstpp\.idea\libraries\KotlinJavaRuntime.xml (created)
  firstpp\.idea\modules.xml (created)
  firstpp\.idea\runConfigurations\main_dart.xml (created)
  firstpp\.idea\workspace.xml (created)
  firstpp\.metadata (created)
  firstpp\analysis_options.yaml (created)
```

```
All done!
In order to run your application, type:

  $ cd firstpp
  $ flutter run

Your application code is in firstpp\lib\main.dart.


c:\Flutter Projects>cd firstpp

c:\Flutter Projects\firstpp>code .
```



firstapp 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.

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

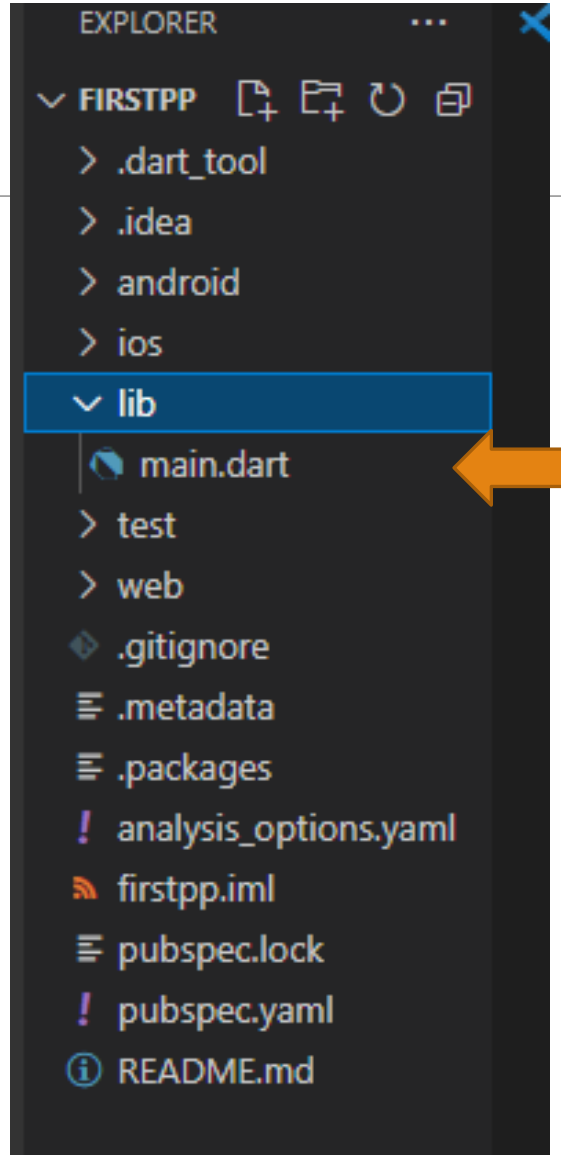
Hot Restarting

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.



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

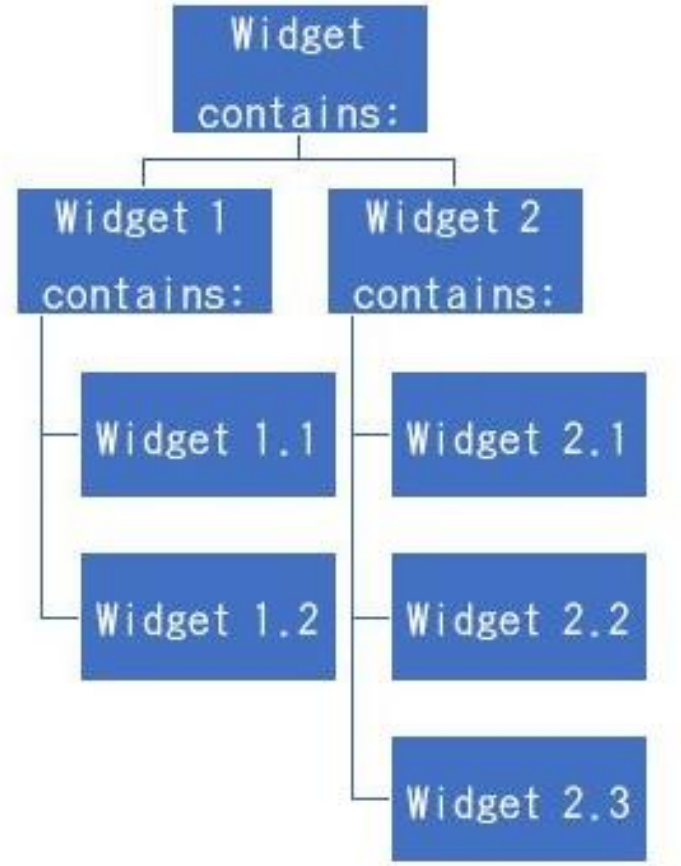
Widgets are the Building Blocks of your UI. Whenever we build a user interface in Flutter, it is composed of Widgets. Putting your widgets together is called Composition. Think of a user interface as a jigsaw. Each widget is a piece of the puzzle

```
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

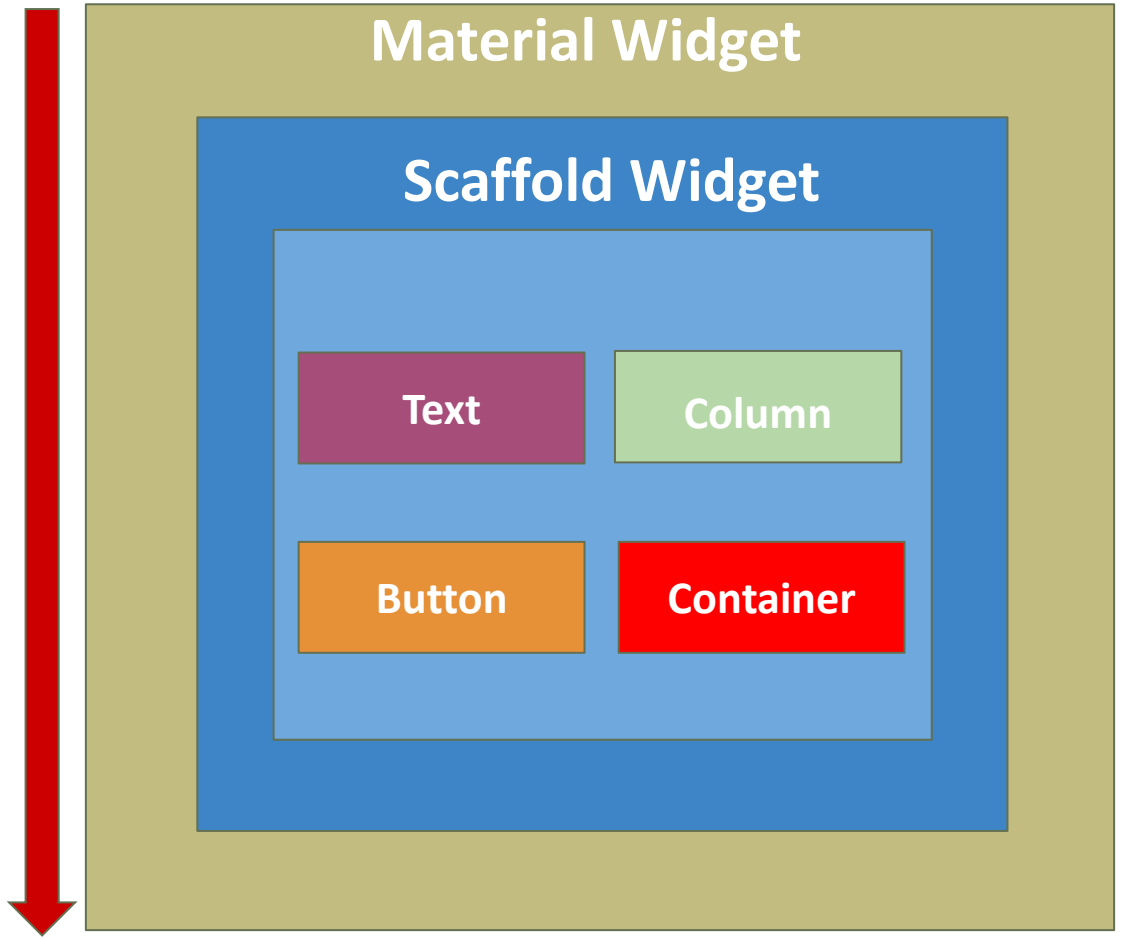
Widget Tree

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

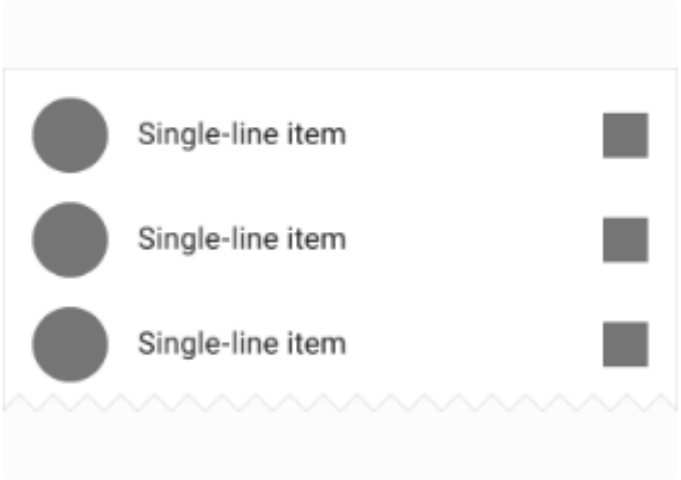


App Widget

Render Tree



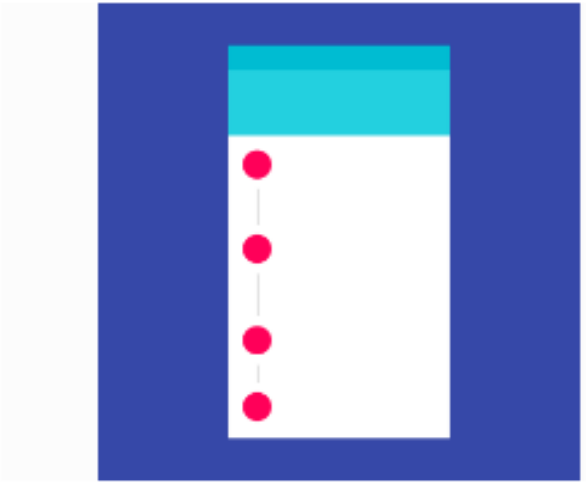
Layout



ListTile

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

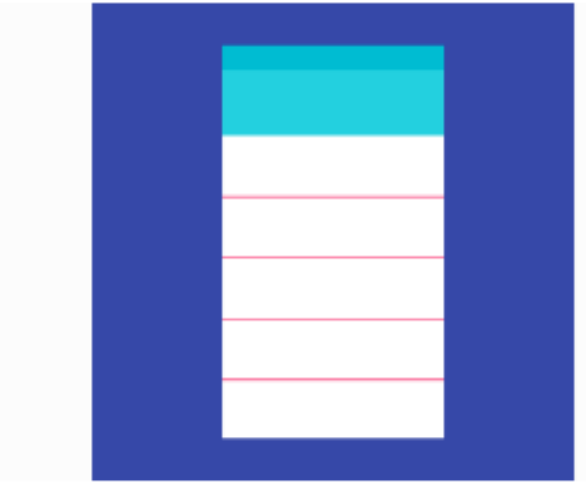
Documentation



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

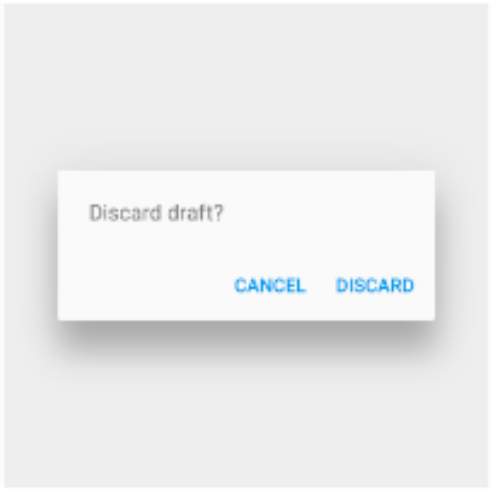
https://flutter.io/widgets/widgetindex/

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

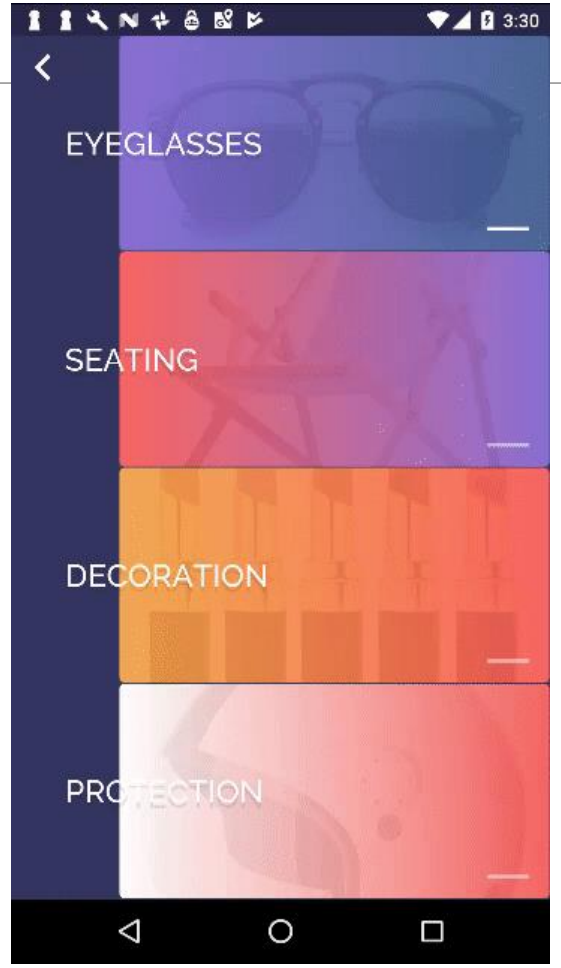
Async

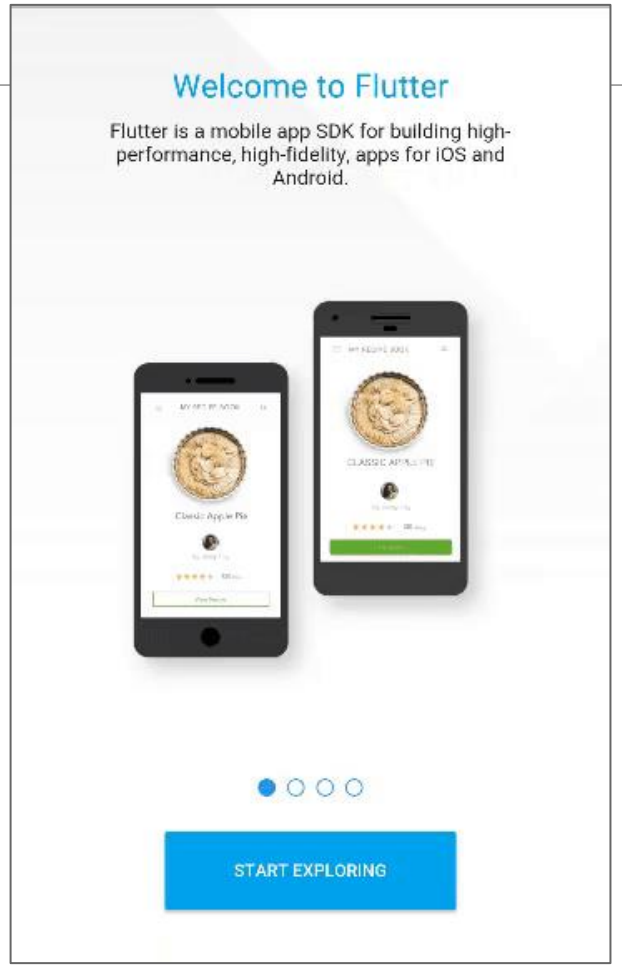
Async patterns to your Flutter application.

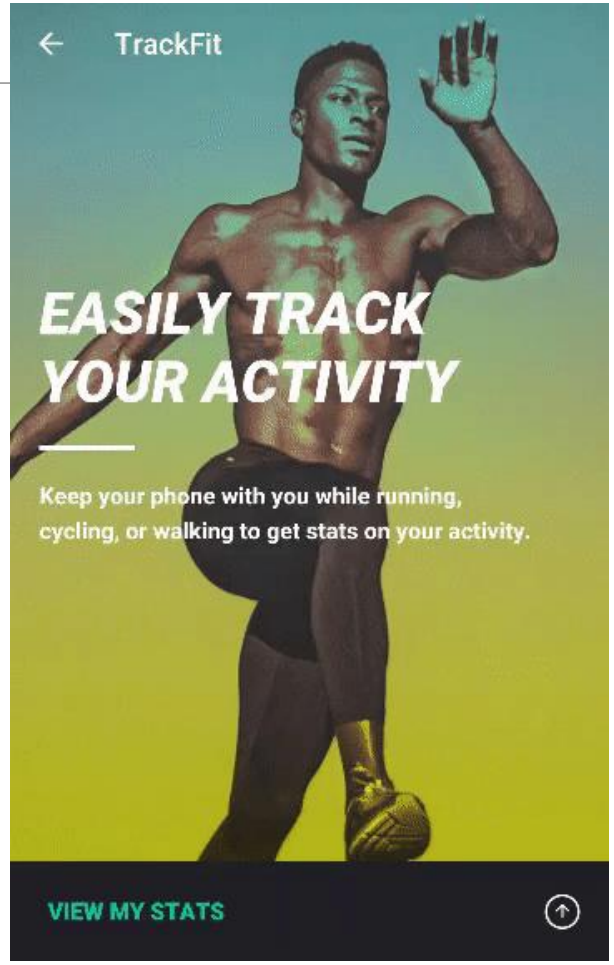
VISIT

https://flutter.io/widgets/

Great looking and fast Widgets







Creating the real HELLOWORLD app

Coding Steps hit here 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

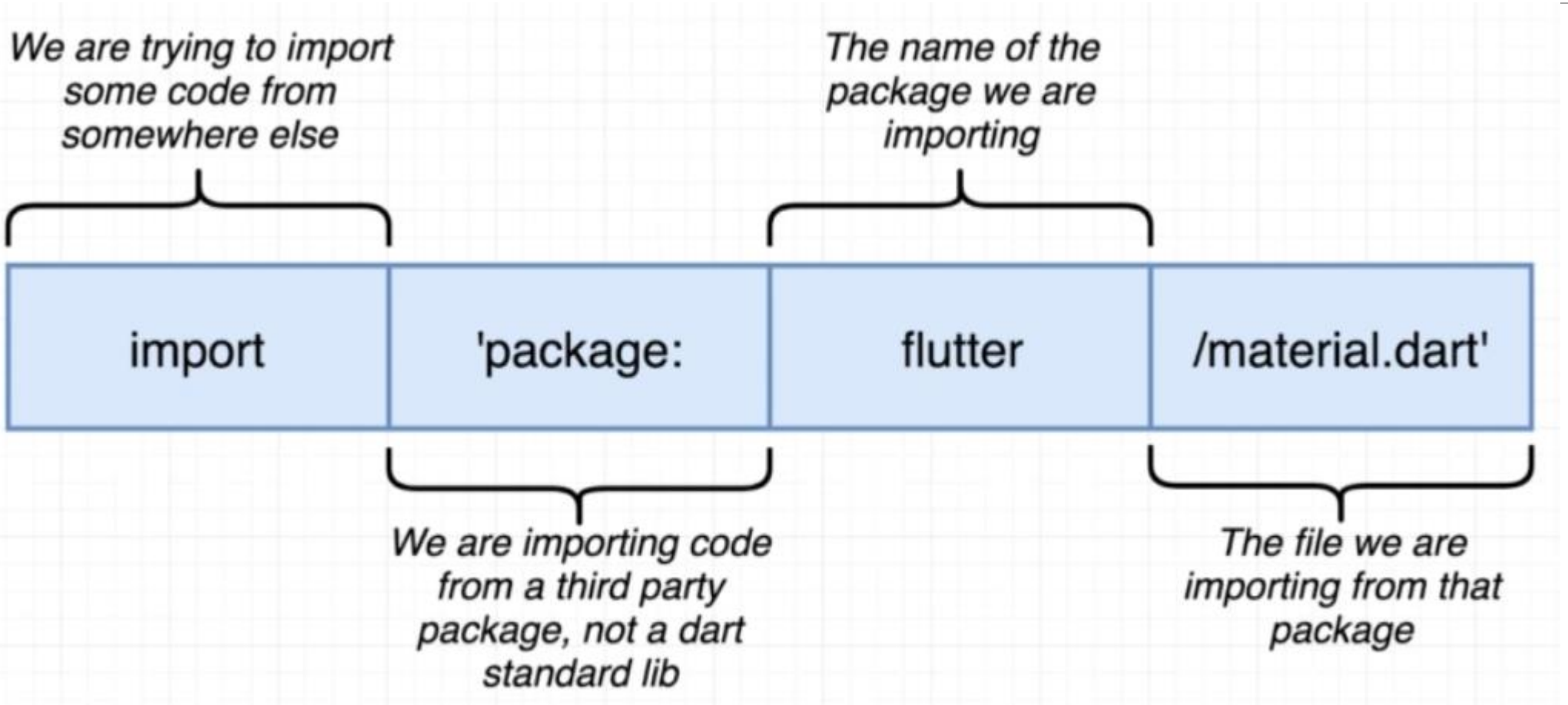
Four step design process

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



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 (Text Widget)

lib > main.dart > ...

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


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


//Create a text widget to show some text on the screen

//Take the widget and get it on the screen

Run | Debug
void main () {
  runApp(Text('Hi there!',textDirection: TextDirection.ltr,));

}

```

Text Widget

Format Text Widget Using **TextStyle**

Change the format of the Text Widget

1. FontSize
2. FontWeight
3. Color

main.dart

lib > main.dart > ...

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



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

Run | Debug
void main () {
//Take the widget and get it on the screen
  runApp(
    //Create a text widget to show some text on the screen
    Text(
      'Hi there!',
      textDirection: TextDirection.ltr,
      style: TextStyle(fontSize: 40.0,
      fontWeight: FontWeight.bold,
      color: Colors.redAccent

      ), // TextStyle

      )); // Text

}


```

Show the Widget on the Screen

main.dart

lib > main.dart > ...

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



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

Run | Debug
void main () {
//Take the widget and get it on the screen
  runApp(
    //Create a text widget to show some text on the screen
    Text(
      'Hi there!',
      textDirection: TextDirection.ltr,
      style: TextStyle(fontSize: 40.0,
      fontWeight: FontWeight.bold,
      color: Colors.redAccent

      ), // TextStyle
      textAlign: TextAlign.center)
      )); // Text

}


```

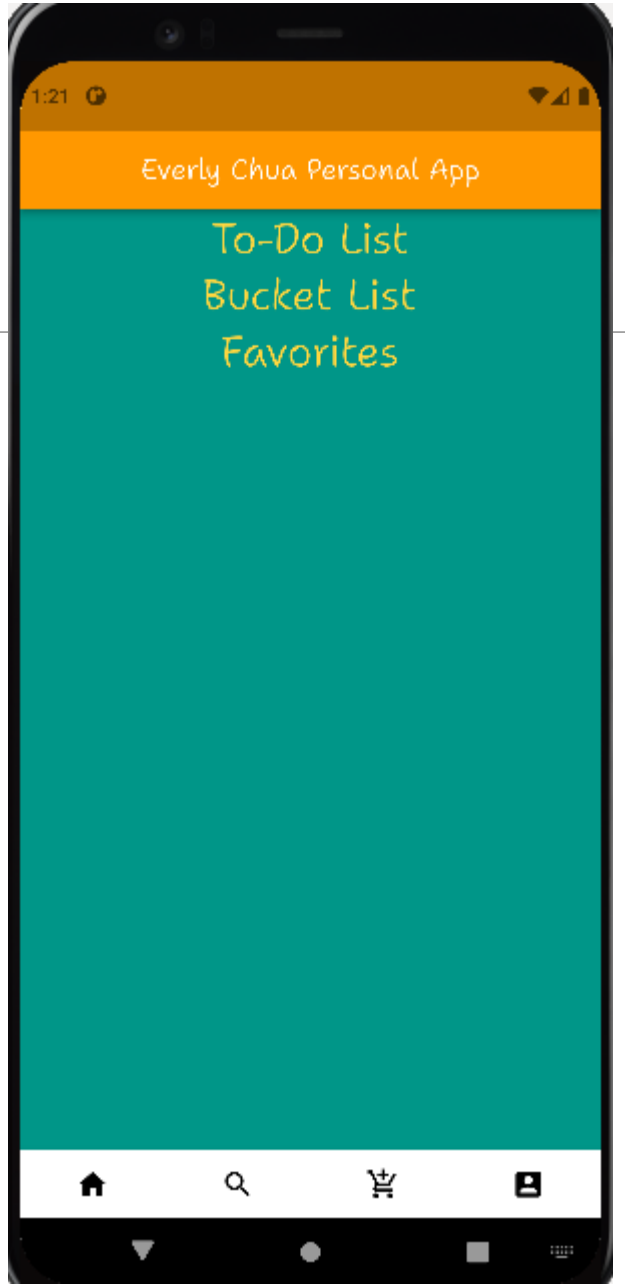
Run the App

Press F5

or

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

Designing UI in Flutter



Widget To Explore

Center

Text

Scaffold

AppBar

BottomNavigationBar

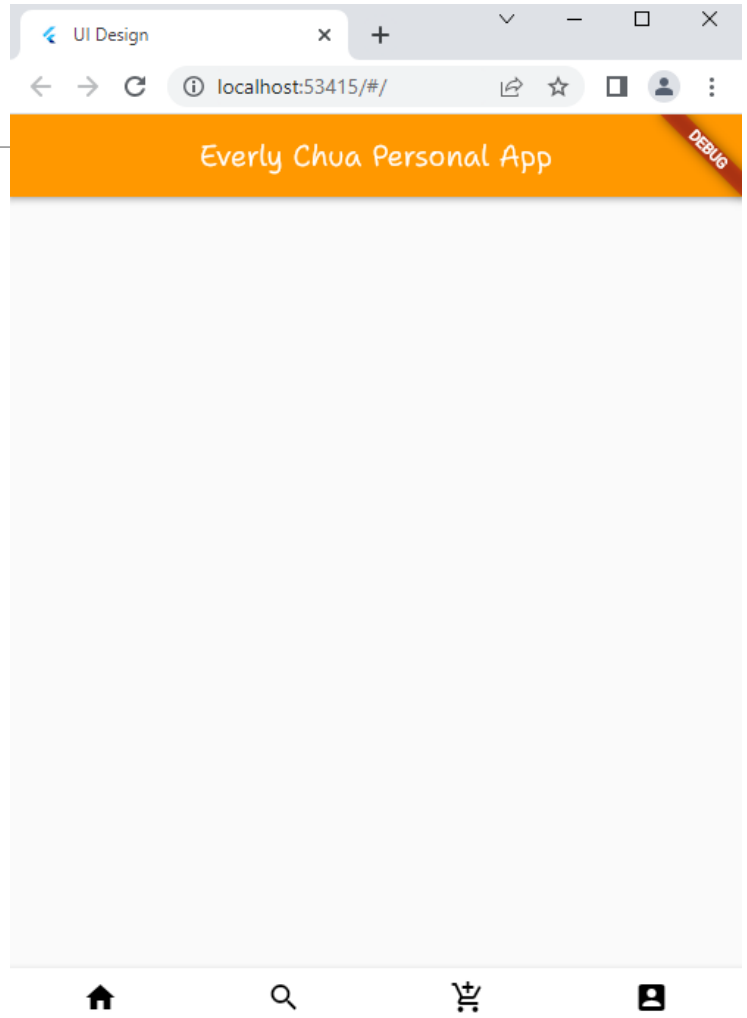
MaterialApp

Material

Scaffold Class

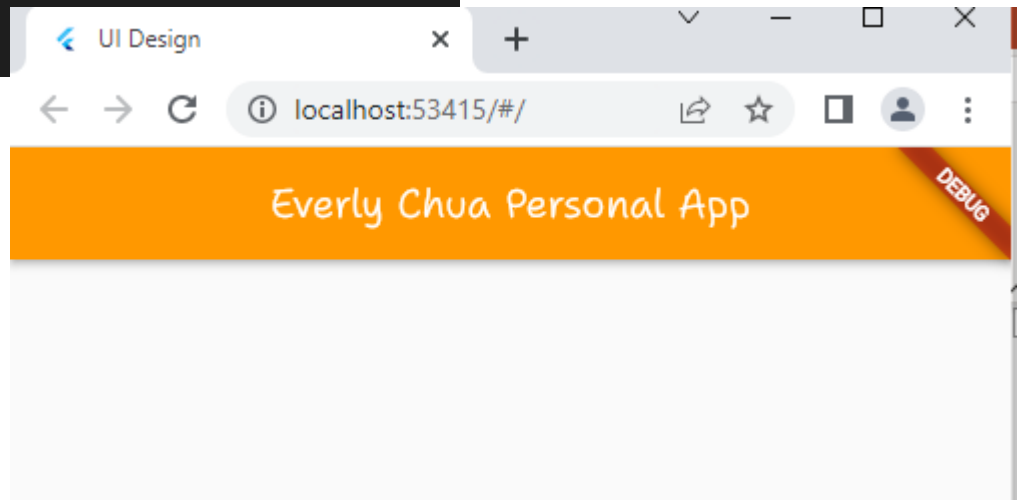
A Scaffold Widget provides a framework which implements the basic material design visual layout structure of the flutter app.

```dart
import 'package:flutter/material.dart';
Run | Debug | Profile
void main(){
  runApp(MaterialApp(title: 'UI Design',
    home: Scaffold( appBar: AppBar(
    title: const Text('Everly Chua Personal App',
    style: TextStyle(fontFamily: 'ShantellSans')), // Text
    backgroundColor: Colors.orange, centerTitle: true,), // AppBar
      bottomNavigationBar: BottomAppBar(
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: [
             IconButton(
                icon: const Icon(Icons.home),
                color: Colors.black,
                onPressed: () {},
              ), // IconButton
             IconButton(
                icon: const Icon(Icons.search),
                color: Colors.black,
                onPressed: () {},
              ), // IconButton
             IconButton(
                icon: const Icon(Icons.add_shopping_cart),
                color: Colors.black,
                onPressed: () {},
              ), // IconButton
              IconButton(
                icon: const Icon(Icons.account_box),
                color: Colors.black,
                onPressed: () {},
              ), // IconButton
            ],
          ), // Row

  ))); // BottomAppBar // Scaffold // MaterialApp
}
```



To remove Debug Banner on App

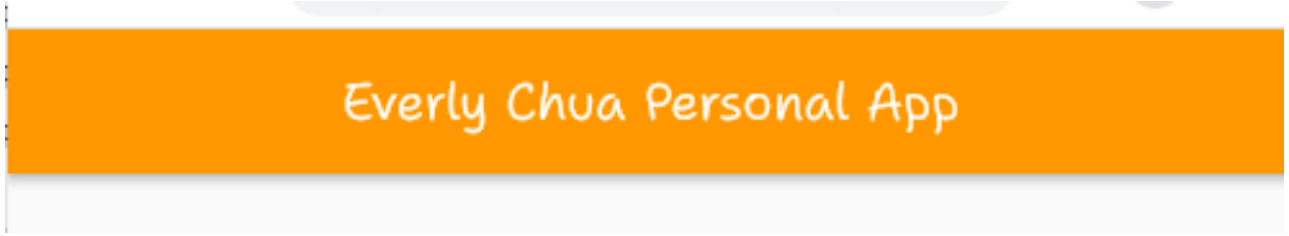
```dart
import 'package:flutter/material.dart';

Run | Debug | Profile
void main(){
  runApp(MaterialApp(title: 'UI Design',
  debugShowCheckedModeBanner: false,
  home: Scaffold( appBar: AppBar(
```

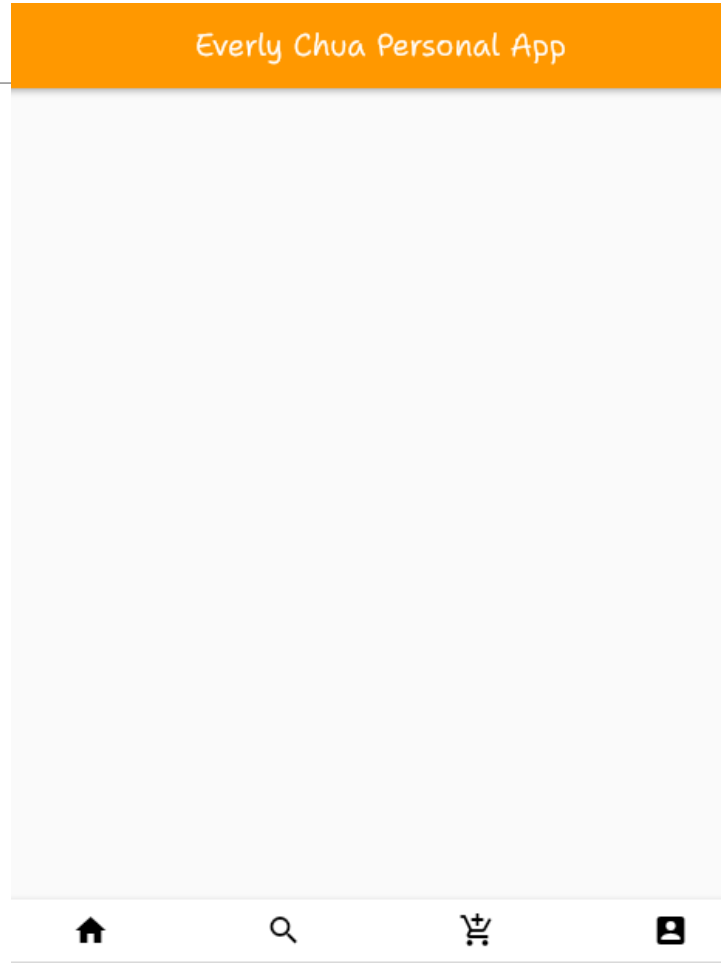


Add debugShowCheckedModeBanner: false,

AppBar

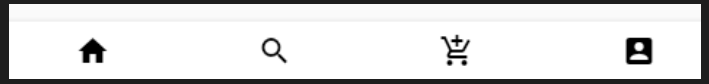
```
home: Scaffold( appBar: AppBar(
  title: const Text('Everly Chua Personal App',
  style: TextStyle(fontFamily: 'ShantellSans')), // Text
  backgroundColor: Colors.orange, centerTitle: true,), // AppBar
```

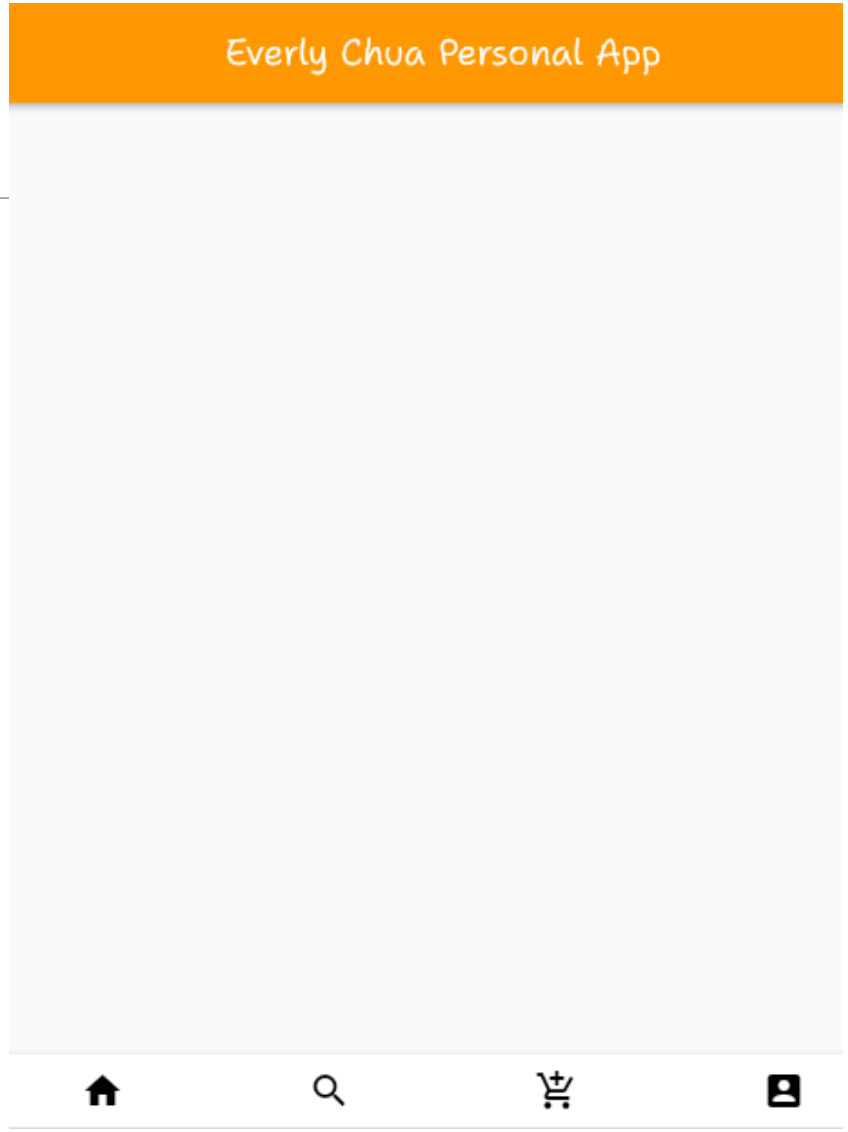




BottomNavigationBar

```
bottomNavigationBar: BottomAppBar(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceAround,
      children: [
       IconButton(
          icon: const Icon(Icons.home),
          color: Colors.black,
          onPressed: () {},
       ), // IconButton
       IconButton(
          icon: const Icon(Icons.search),
          color: Colors.black,
          onPressed: () {},
       ), // IconButton

        IconButton(
          icon: const Icon(Icons.add_shopping_cart),
          color: Colors.black,
          onPressed: () {},
        ), // IconButton
        IconButton(
          icon: const Icon(Icons.account_box),
          color: Colors.black,
          onPressed: () {},
        ), // IconButton
      ],
    ), // Row
  ), // BottomAppBar
```



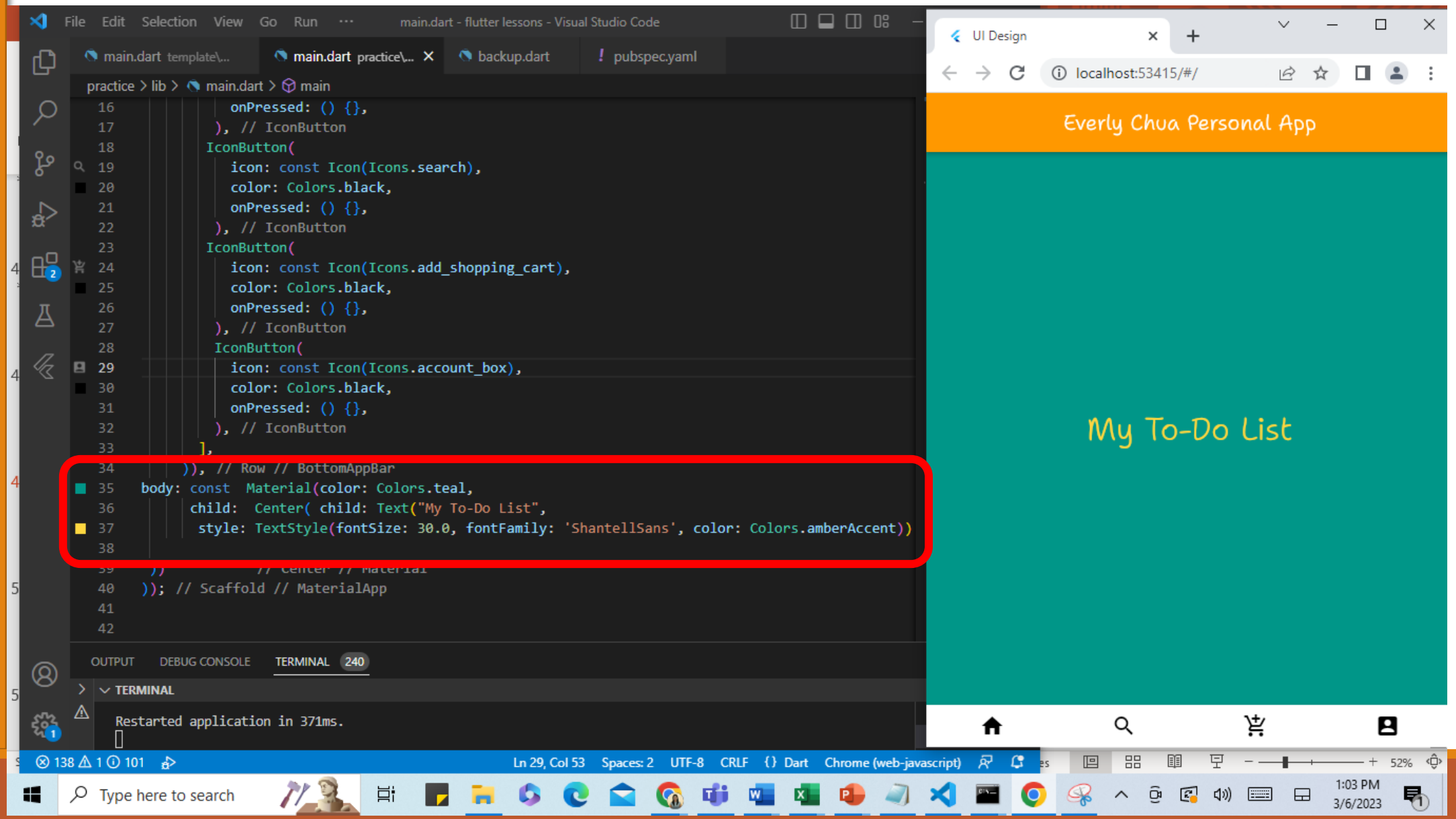


MaterialApp and Materials

MaterialApp is a widget **to create widgets to design applications in Flutter**. The Material app has several properties. Some of them are title, home, theme, color that helps developer design their app

Material, on the other hand, a widget used to define a UI element respecting Material rules such as what elevation is, widget shape, and stuff. It is reused by many material widgets such as App bar or Card or Floating Button.

Using MaterialApp and Material

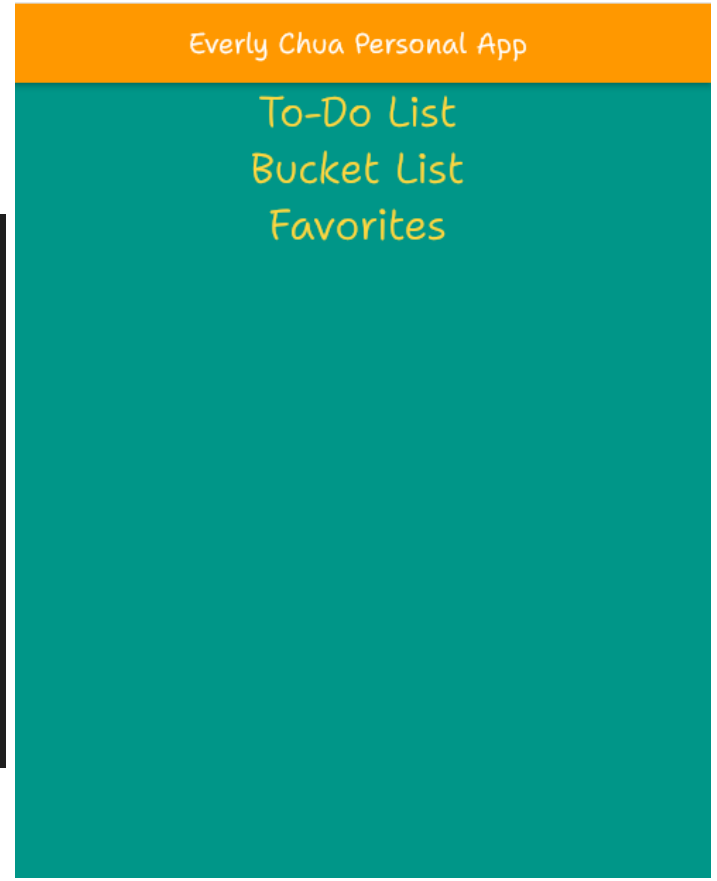


Columns Widget

A widget that displays its children in a vertical array.

To cause a child to fill the available vertical space,

```
    body:  Material(color: Colors.teal,
        child: Center(
       child: Column(children: const <Widget>[
          Text("To-Do List",
          style: TextStyle(fontSize: 30.0, fontFamily: 'ShantellSans', color: Colors.amberAccent)), // Text

          Text("Bucket List",
          style: TextStyle(fontSize: 30.0, fontFamily: 'ShantellSans', color: Colors.amberAccent)), // Text

           Text("Favorites",
           style: TextStyle(fontSize: 30.0, fontFamily: 'ShantellSans', color: Colors.amberAccent)) // Text
      ]) // <Widget>[] // Column
      ))          // Center // Material
    ))); // Scaffold // MaterialApp
}
```



Containers

A container first surrounds the child with padding (inflated by any borders present in the decoration) and then applies additional constraints to the padded extent (incorporating the width and height as constraints, if either is non-null). The container is then surrounded by additional empty space described from the margin.

By default the container width and height is based on its parent widget which is the MaterialApp

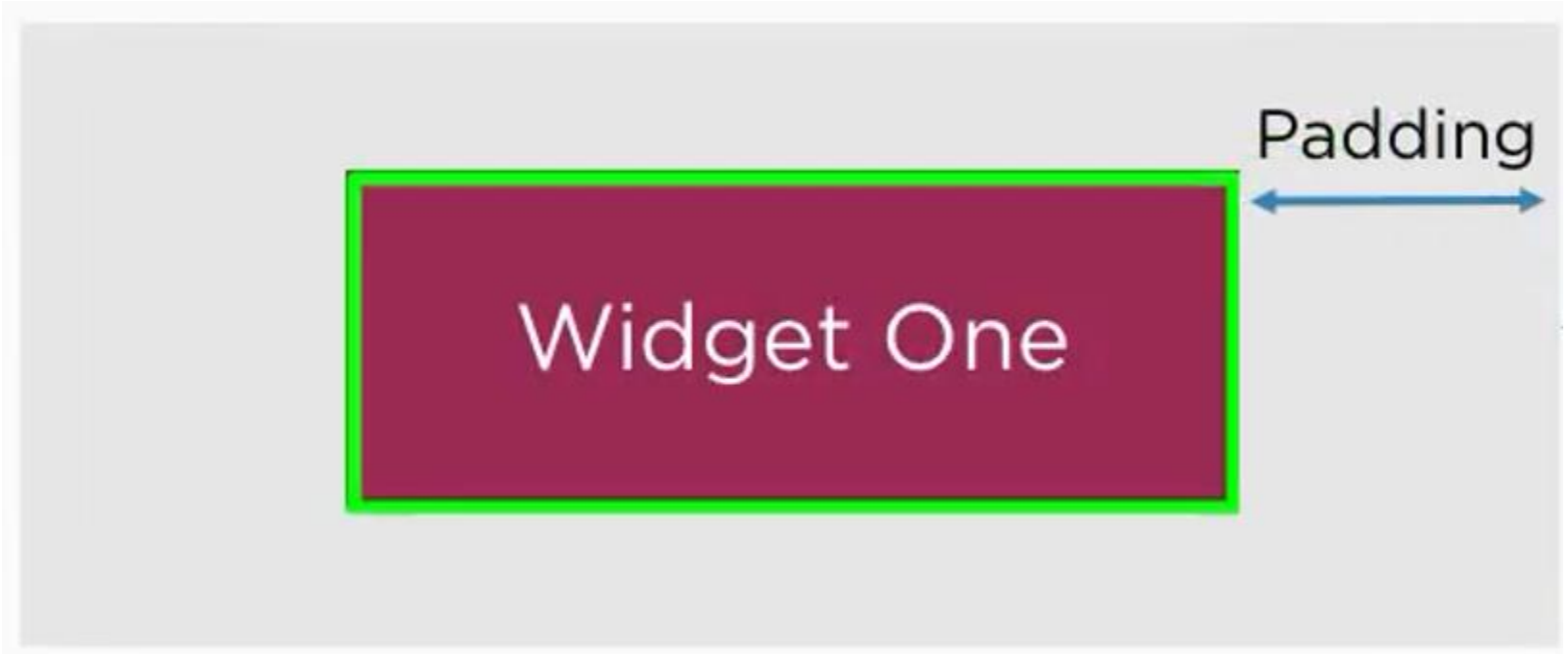
Properties: margin, height, width, alignment, borderRadius, decoration

https://api.flutter.dev/flutter/widgets/Container-class.html

Margins vs. Padding

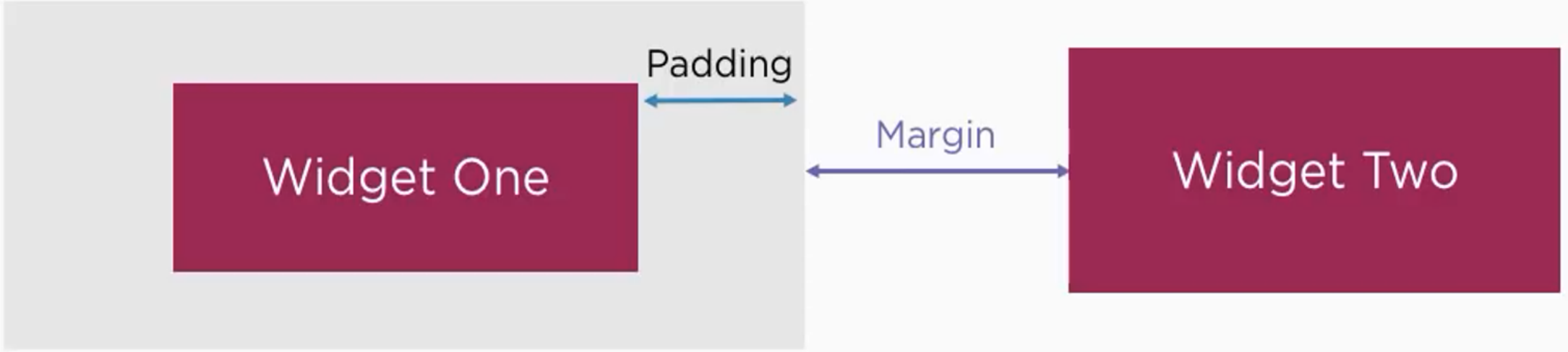
Margin is the distance between two widgets

Margins vs. Padding



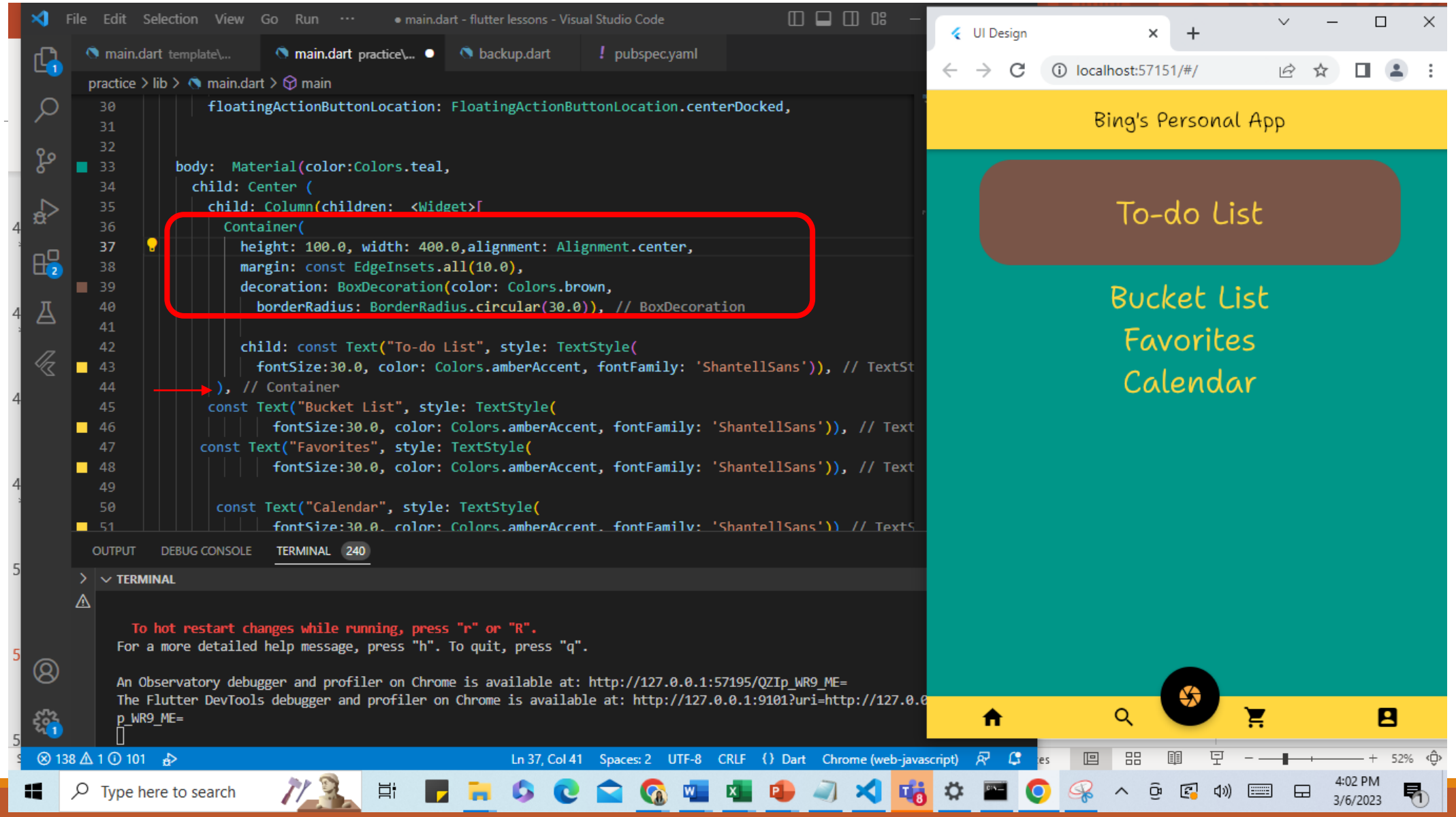
Boundary inside the widget

Margins vs. Padding

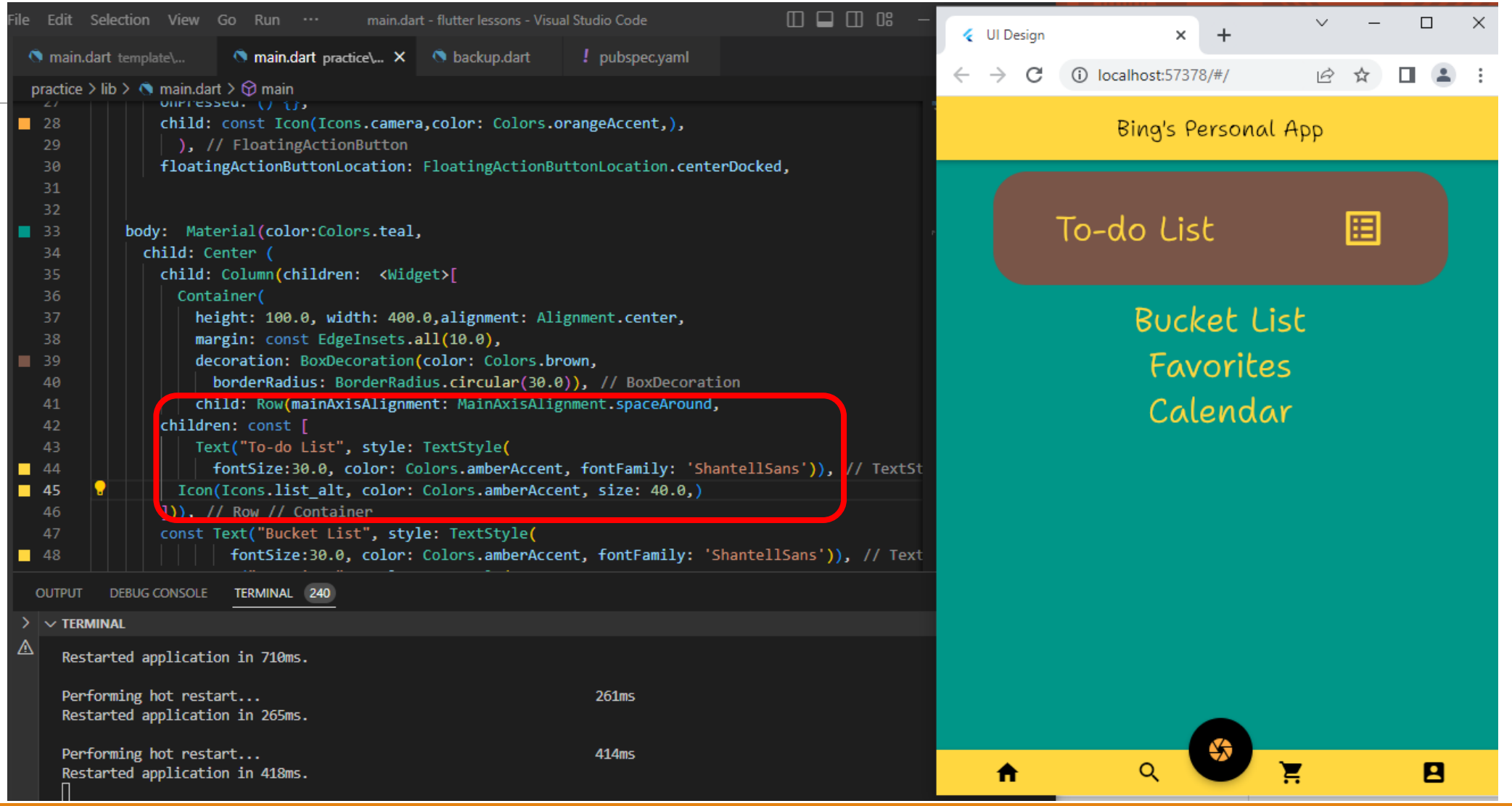



```
EdgeInsets.All(20.0)
EdgeInsets.Only(right: 20.0)
```

Containers


File Edit Selection View Go Run
main.dart - flutter lessons - Visual Studio Code
main.dart template\...
main.dart practice\...
backup.dart
pubspec.yaml
practice > lib > main.dart > main
floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
body: Material(color:Colors.teal,
child: Center (
child: Column(children: <Widget>[
Container(
height: 100.0, width: 400.0,alignment: Alignment.center,
margin: const EdgeInsets.all(10.0),
decoration: BoxDecoration(color: Colors.brown,
borderRadius: BorderRadius.circular(30.0)), // BoxDecoration
child: const Text("To-do List", style: TextStyle(
fontSize:30.0, color: Colors.amberAccent, fontFamily: 'ShantellSans')), // TextSt
), // Container
const Text("Bucket List", style: TextStyle(
fontSize:30.0, color: Colors.amberAccent, fontFamily: 'ShantellSans')), // Text
const Text("Favorites", style: TextStyle(
fontSize:30.0, color: Colors.amberAccent, fontFamily: 'ShantellSans')), // Text
const Text("Calendar", style: TextStyle(
OUTPUT DEBUG CONSOLE TERMINAL 240
TERMINAL
To hot restart changes while running, press "r" or "R".
For a more detailed help message, press "h". To quit, press "q".
An Observatory debugger and profiler on Chrome is available at: http://127.0.0.1:57195/QZIp_WR9_ME=
The Flutter DevTools debugger and profiler on Chrome is available at: http://127.0.0.1:9101?uri=http://127.0.0.1:57195/QZIp_WR9_ME=
Ln 37, Col 41 Spaces: 2 UTF-8 CRLF Dart Chrome (web-javascript)
UI Design
localhost:57151/#/
Bing's Personal App
To-do List
Bucket List
Favorites
Calendar
Type here to search
4:02 PM
3/6/2023

ROW widget A widget that displays its children in a horizontal array.



```
body:  Material(color:Colors.teal,
  child: Center (
    child: Column(children:  <Widget>[
      Container(
        height: 100.0, width: 400.0,alignment: Alignment.center,
        margin: const EdgeInsets.all(10.0),
        decoration: BoxDecoration(color: Colors.brown,
          borderRadius: BorderRadius.circular(30.0)), // BoxDecoration
        child: Row(mainAxisAlignment: MainAxisAlignment.spaceAround,
         children: const [
          Text("To-do List", style: TextStyle(
          fontSize:30.0, color: Colors.amberAccent, fontFamily: 'ShantellSans')), // TextS
          Icon(Icons.list_alt, color: Colors.amberAccent, size: 40.0,),
                |
    ])), // Row // Container
```

End of Presentation