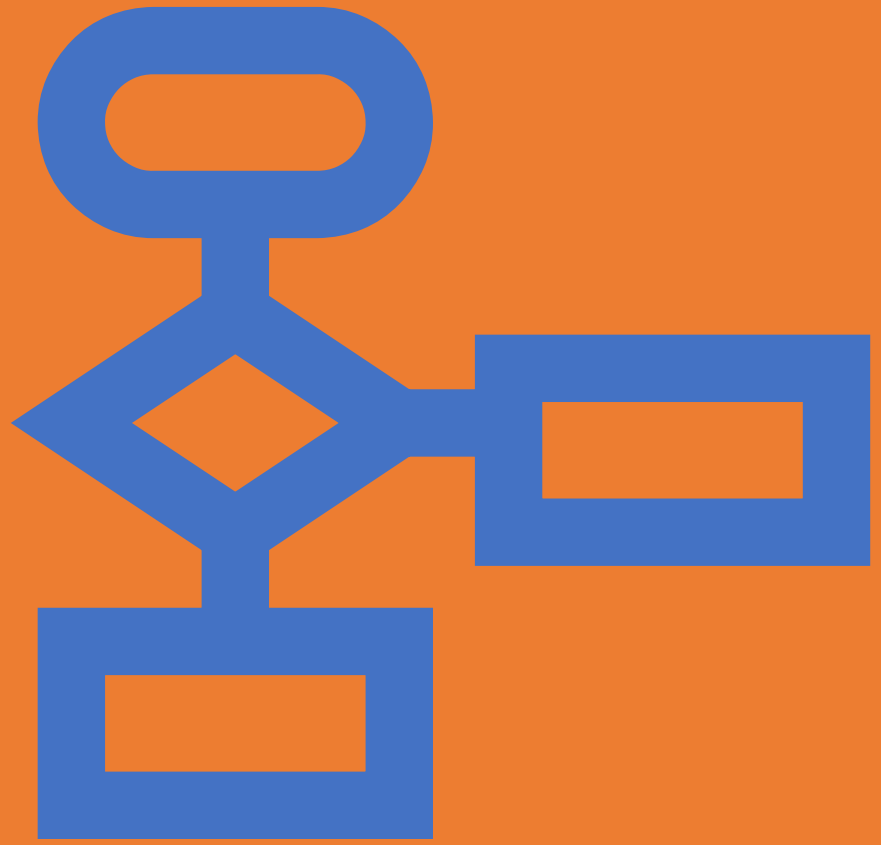
Flutter Project Structure and Widgets

Written by Thapanapong Rukkanchanunt

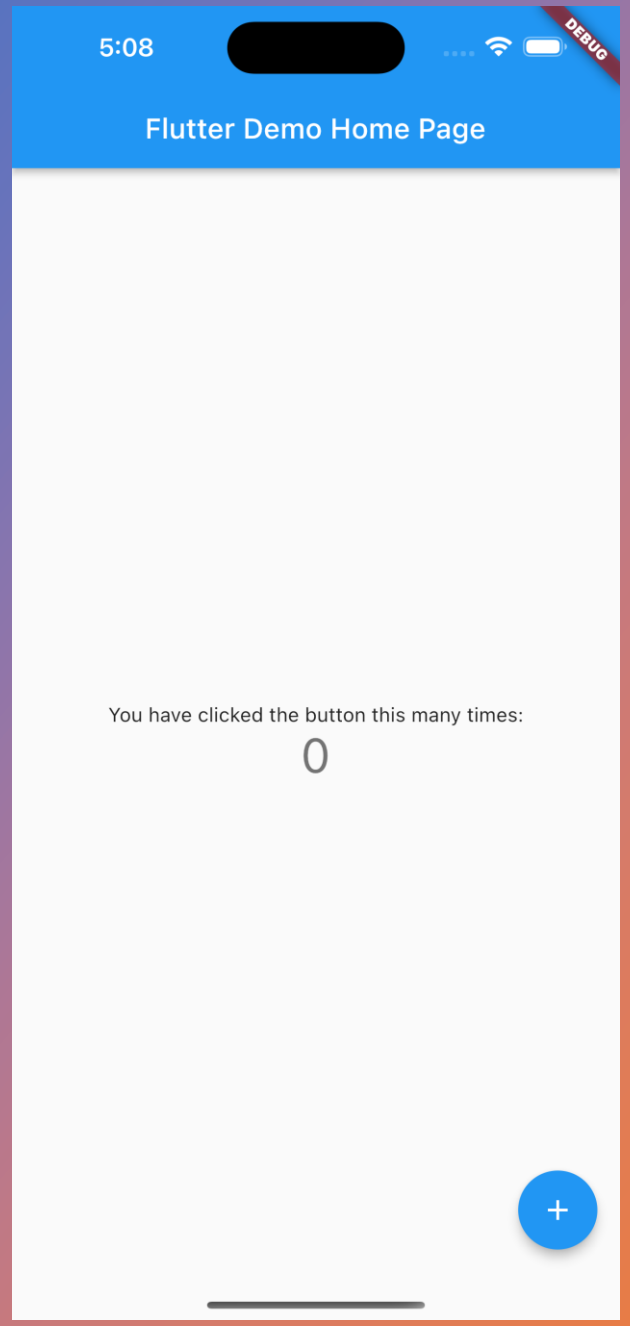


Outline

- First App
- Widget
- Material App
- Scaffold
- Visible Widget
- Assets and Images
- Hot Reload vs Hot Restart vs Full Restart
- Stateless Widget

Let’s build our first app

1. Open Android Studio
2. Click Plugins, search for Flutter, and then install.
3. Click New Flutter Project, and type the project name
4. After the project is created, locate the main toolbar, and click Open Android Simulator or iOS Simulator
5. Once the simulator is loaded completely, click Run to start the app



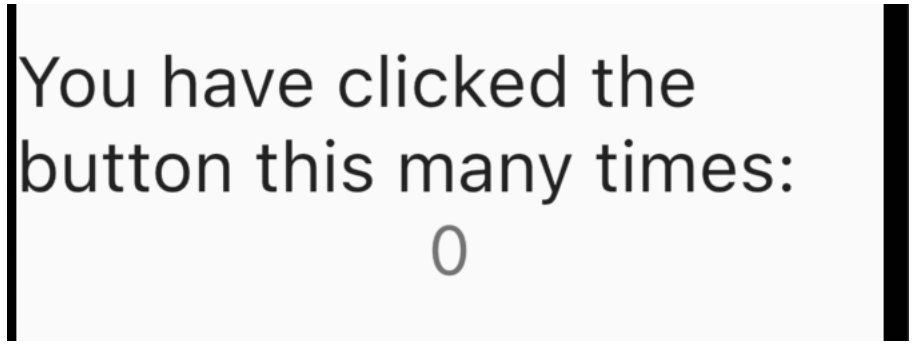
Hooray! Your first app is finished

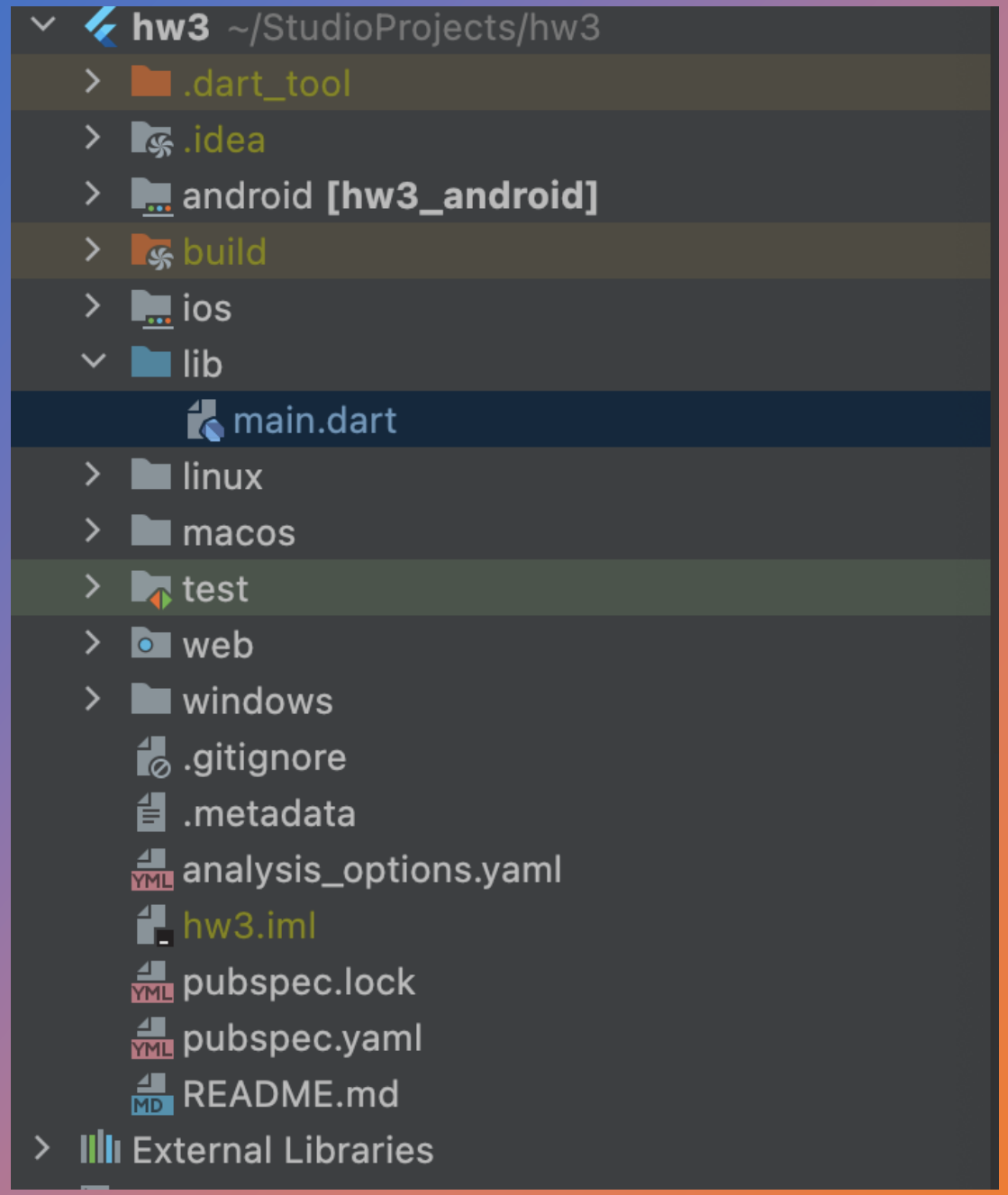
Your app contains: (in main.dart)

- main function
- MyApp class
- MyHomePage class
- _MyHomePageState class

Edit and Hot Reload

- Your main function will be in lib/main.dart
- Change the word “pushed” to “clicked” in Build function of _MyHomePageState class
- Change the font size by adding the following line below the text
  - style: TextStyle(fontSize: 35),
- Click “Hot Reload”





Project Folders

- When we create new Flutter project, there are many folders, files generated by the Android Studio
- We write codes in /lib
- Platform specific codes are in /ios, /android, /linux, /macos, /web, /windows, etc.
- We write tests in /test

```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: 'Assignment #3 Hello User',
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),  // ThemeData
      home: const MyHomePage(title: 'Assignment #3 Hello User'),
    );  // MaterialApp
  }
}
```

Widget

- Everything in Flutter is Widget
- Widget is element you see on screen
  - Text, Container, etc.
- You can customize configuration such as color, shape.
- Widget can be stateful or stateless
  - Stateful widget is dynamic
  - Stateless widget never changes
- Function setState() will call build function of all stateful widget

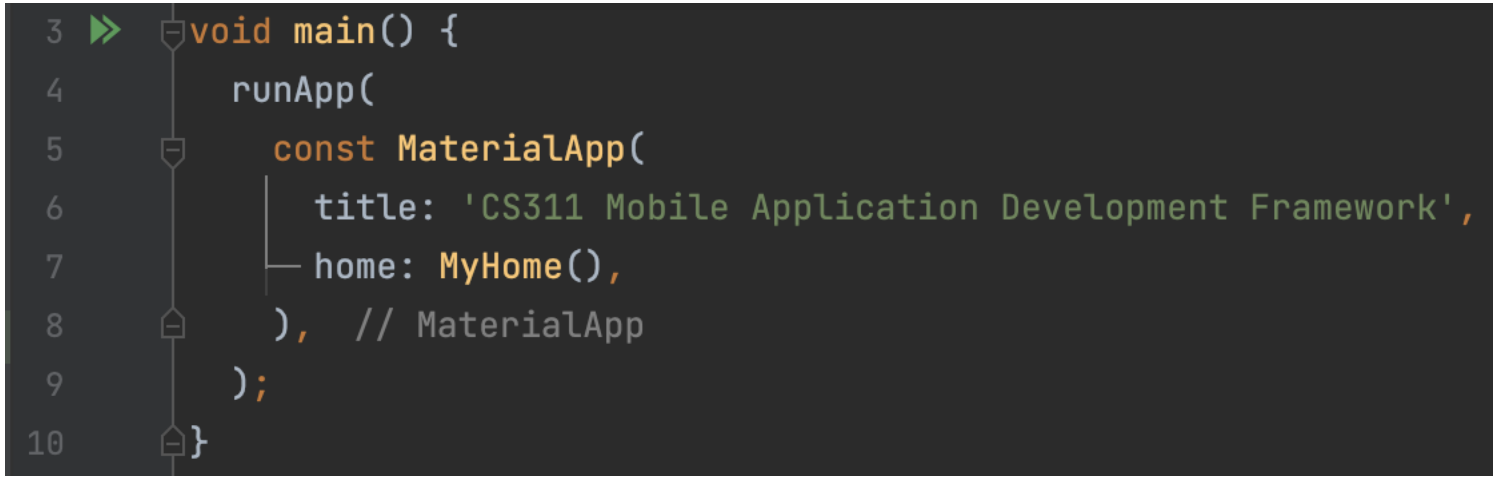
```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),  // AppBar
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text(
            greetText(),
            style: Theme.of(context).textTheme.displayMedium,
          ),  // Text
        ],  // <Widget>[]
      ),  // Column
    ),  // Center
    floatingActionButton: FloatingActionButton(
      onPressed: _loginUser,
      tooltip: 'login',
      child: const Icon(Icons.login),
    ),  // FloatingActionButton
  );  // Scaffold
}
}
```

Widget Element

- When writing a Widget class, we inherit from either StatefulWidget or StatelessWidget
- The main function is build whose parameters describe the widget
- In the IDE, we can see widget tree
  - The root is the widget in the argument of runApp function

Material App

- Flutter provides several widgets that help you build apps that follow Material Design
- A Material app starts with the MaterialApp widget, which builds several useful widgets at the root of your app such as Navigator
- Using the MaterialApp widget is highly recommended



pubspec.yml

- To use Material Design, make sure that uses-material-design is true in pubspec.yml
- pubspec.yml has information about your app configuration

```
54  # The following section is specific to Flutter packages.
55  flutter:
56
57    # The following line ensures that the Material Icons font is
58    # included with your application, so that you can use the icons in
59    # the material Icons class.
60    uses-material-design: true
```

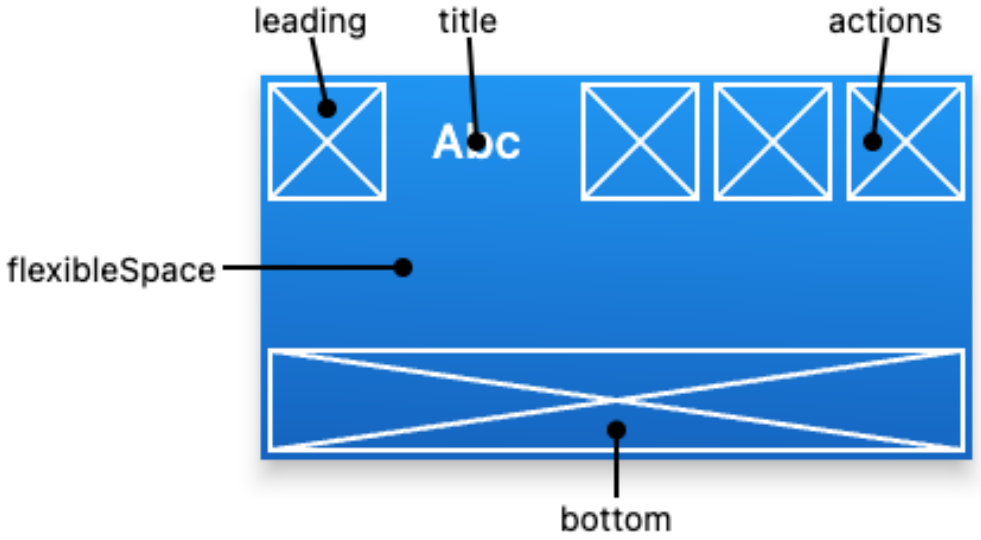
Scaffold

```
home: Scaffold(
  appBar: AppBar(...),  // AppBar
  floatingActionButton: const FloatingActionButton(...),
  body: const Center(child: Text('Body of your App')),
  bottomNavigationBar: BottomNavigationBar(...),  // Bot
),  // Scaffold
```

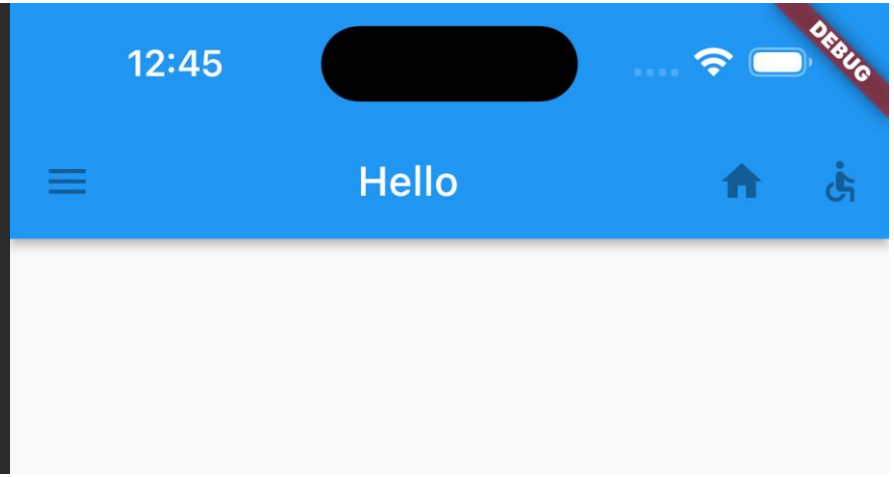
- Scaffold is a layout structure.
- It is one page view in your app.
- appBar: a horizontal bar at the top of the app
  - Use AppBar class to specific the bar
- floatingActionButton: a circular button at the bottom right
- body: an area in middle of your app
- bottomNavigationBar: a horizontal bar at the bottom of the app
- And many more to come

appBar example

- Commonly used named parameters are leading, title and actions



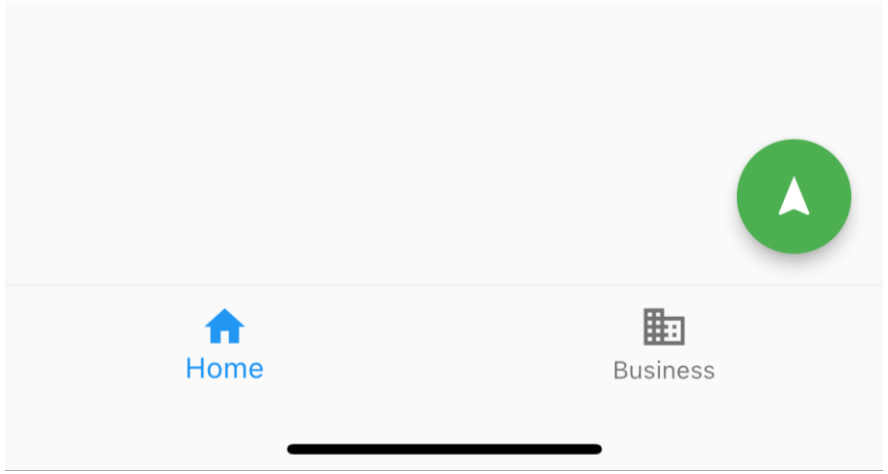

```
appBar: AppBar(
  leading: const IconButton(onPressed: null, icon: Icon(Icons.menu),),
  title: const Center(child: Text('Hello')),
  actions: const [
    IconButton(onPressed: null, icon: Icon(Icons.home)),
    IconButton(onPressed: null, icon: Icon(Icons.accessible)),
  ],
),  // AppBar
```



bottomNavigationBar Example

- items must have at least 2 elements

```
bottomNavigationBar: BottomNavigationBar(
  items: const [
    BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Home'),
    BottomNavigationBarItem(icon: Icon(Icons.business), label: 'Business'),
  ],
),  // BottomNavigationBar
```



Text

- The Text 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.
- Default parameter is the text itself (no need for named parameter)

```
child: Text(
  'Body of your App',
  style: TextStyle(fontWeight: FontWeight.bold),
)  // Text
```

Container

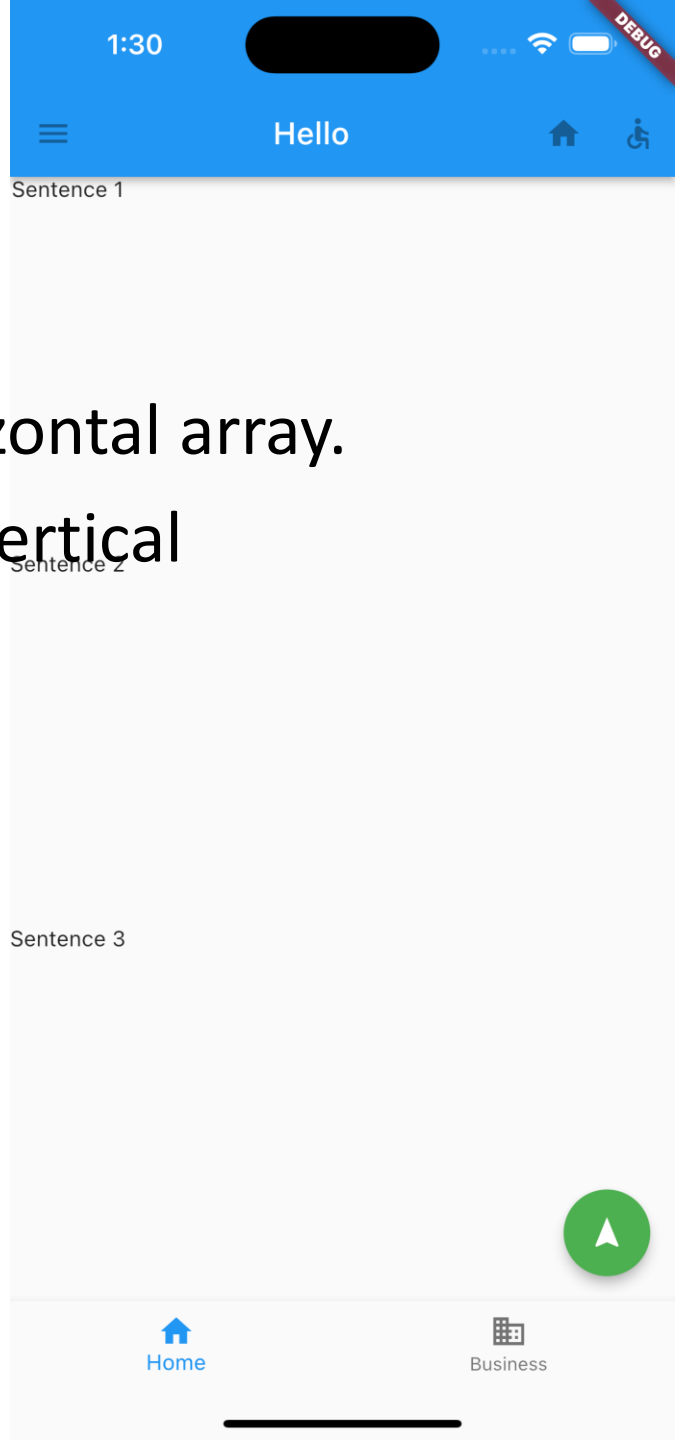
- Container is a widget class that allows you to customize its child widget.
- Use a Container when you want to add padding, margins, borders, or background color, to name some of its capabilities.

```
child: Container(
  margin: const EdgeInsets.all(10.0),
  color: Colors.red,
  width: 500.0,
  height: 250.0,
  transform: Matrix4.rotationZ(0.1),
),  // Container
```

Row and Column

- Row is a widget that displays its children in a horizontal array.
- Column is a widget that displays its children in a vertical array.
- Both have children as their named parameter

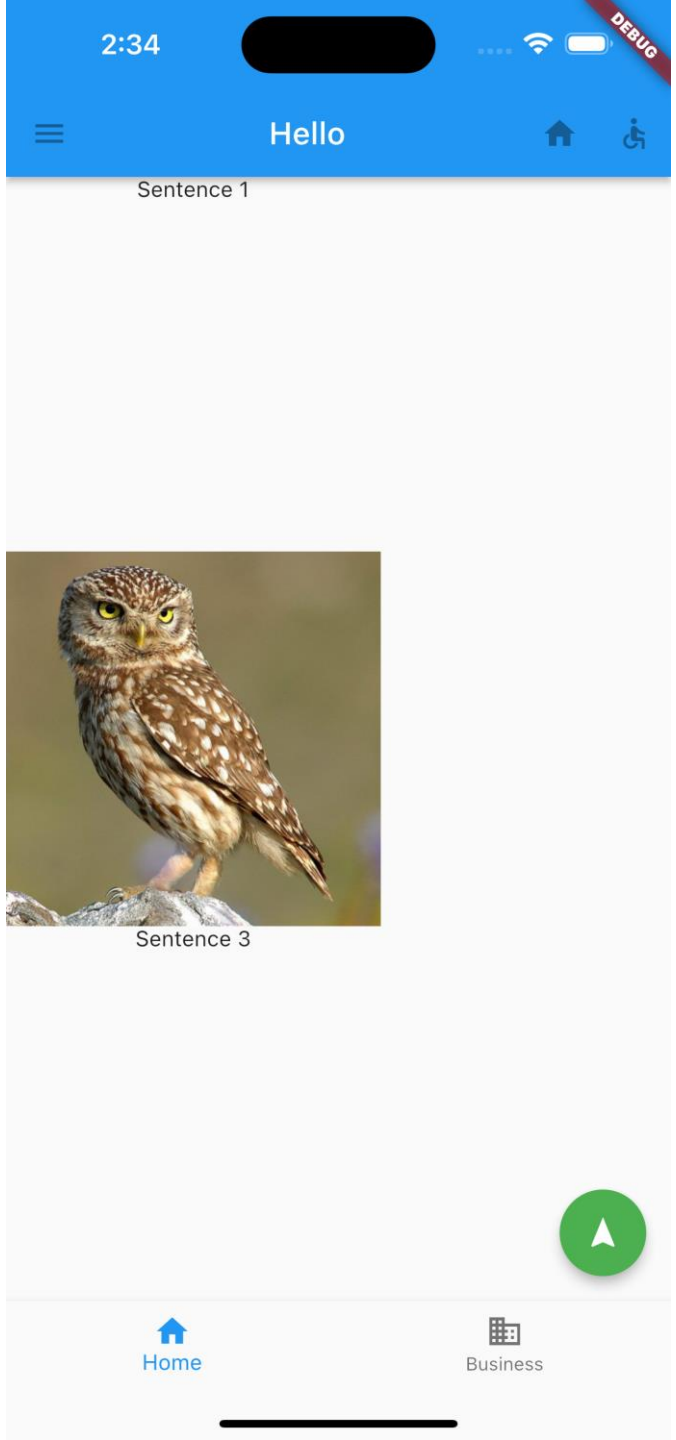
```
body: Column(
  children: const [
    Expanded(child: Text('Sentence 1'),),
    Expanded(child: Text('Sentence 2'),),
    Expanded(child: Text('Sentence 3'),),
  ],
),  // Column
```



Image

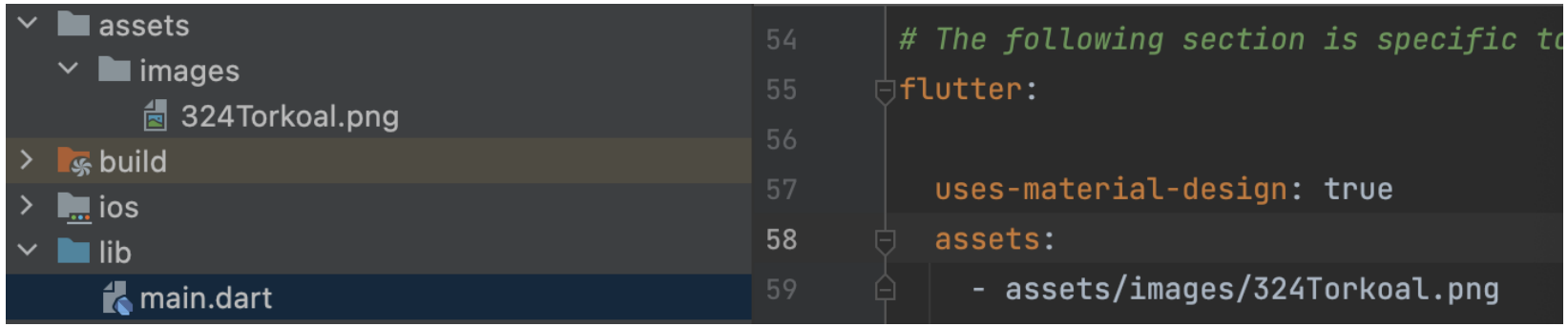
- Image class is a widget that show an image
- Image can come from various source
  - Image.new, for obtaining an image from an ImageProvider.
  - Image.asset, for obtaining an image from an AssetBundle using a key.
  - Image.network, for obtaining an image from a URL.
  - Image.file, for obtaining an image from a File.
  - Image.memory, for obtaining an image from a Uint8List.

```
child: Image(
  image: NetworkImage('https://flutter.github.io/assets-for-api-docs/assets/widgets/owl.jpg'),
),  // Image
```



Assets and Images

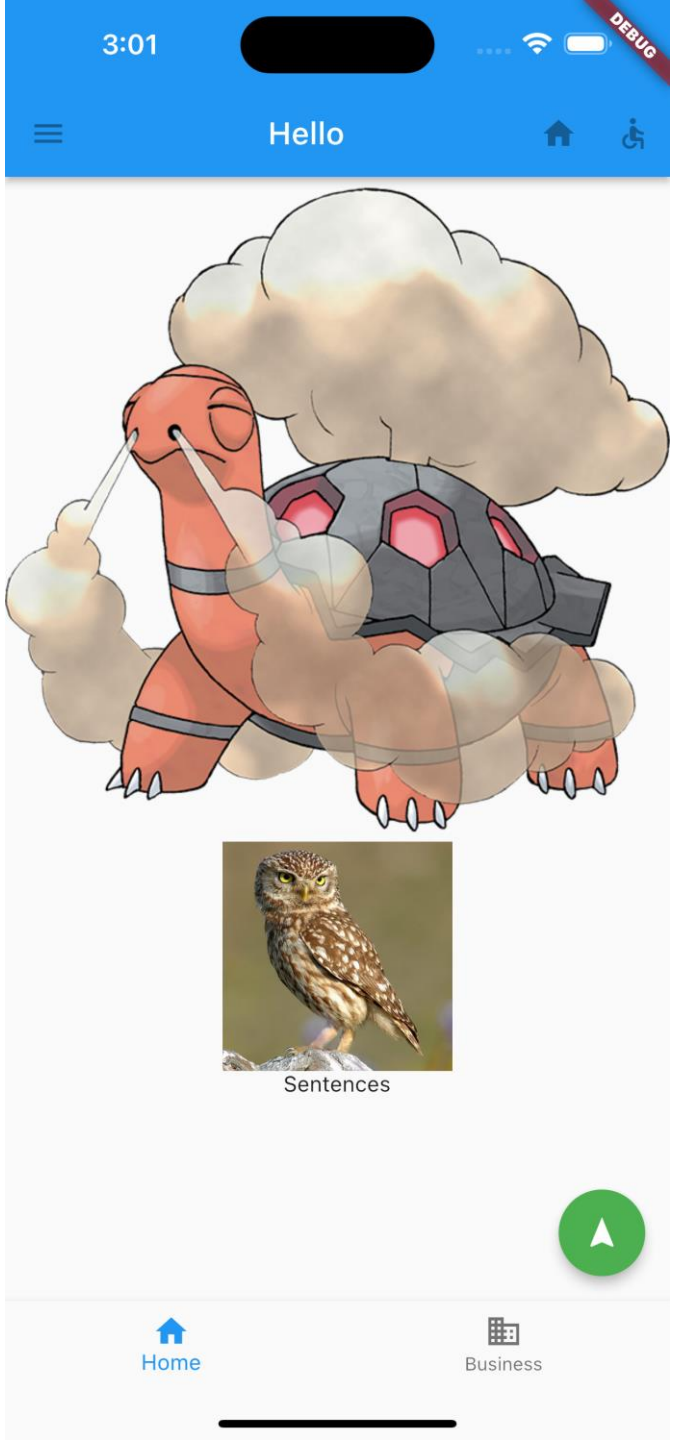
- Flutter uses the pubspec.yaml file, located at the root of your project, to identify assets required by an app.
- It is advised to create assets/images/ directory in your project and put all assets there.
- Then edit pubspec.yaml to include that directory
- Hot Restart is required to reload assets.



Display Image from Assets

- Provide a path from your project root to the image
- You can adjust size using scale, height, width, fit
- You can also blend image using color and colorBlendMode

```
body: Column(
  children: [
    Image.asset('assets/images/324Torkoal.png'),
    const Expanded(
      child: Image(
        image: NetworkImage('https://flutter.githu
      ),  // Image
    ),  // Expanded
    const Expanded(child: Text('Sentences'),),
  ],
),  // Column
```



Hot Reload vs Hot Restart vs Full Restart

- Hot reload loads code changes into the VM and re-builds the widget tree, preserving the app state; it doesn't rerun main() or initState(). All class objects with build() will be re-executed.
- Hot restart loads code changes into the VM, and restarts the Flutter app, losing the app state.
- Full restart restarts the iOS, Android, or web app. This takes longer because it also recompiles the Java / Kotlin / ObjC / Swift code. On the web, it also restarts the Dart Development Compiler.
- Up until now we have not created a class with build function. Let's do that.

MyHome class

- Let’s refactor Home screen into its own class
- Create a class called MyHome that extends StatelessWidget
- Then, write constructor and build() (IDE can help)
- Copy entire Scaffold to the return value of build()
- Now do hot restart
- Edit some texts and try hot reload

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(...),  // AppBar
      floatingActionButton: const FloatingActionButton(...),
      body: Column(...),  // Column
      bottomNavigationBar: BottomNavigationBar(...),  // Bot
    );  // Scaffold
  }

}
```

In-Class Exercise

- Follow “Your first Flutter app” codelab
  - https://codelabs.developers.google.com/codelabs/flutter-codelab-first#0
- Step 2 can be skipped if you are using Lab computer.
- Once you finished, inform staff.