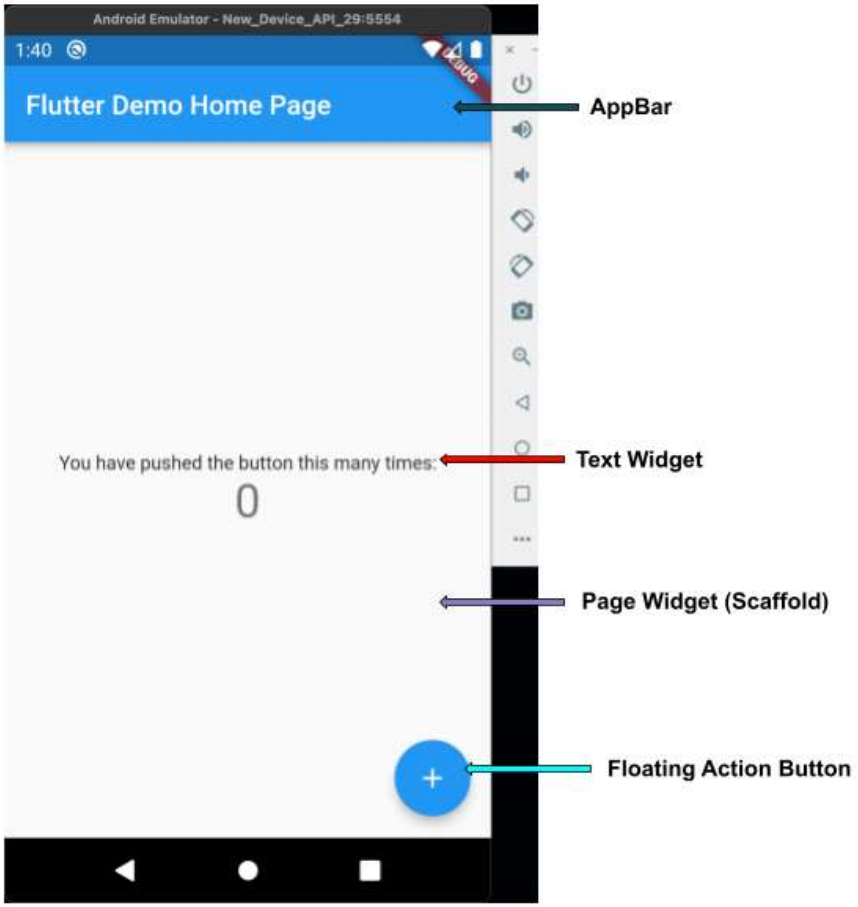
Flutter and Widgets

The Flutter UI is implemented by using widgets from a modern reactive framework. Flutter uses its own rendering engine to draw widgets.



What is a widget?

- Widgets can be compared to LEGO blocks; by adding blocks together, you create an object, and by adding different kinds of blocks, you can alter the look and behavior of the object.

- **Widgets are the building blocks of a Flutter app, and each widget is an immutable declaration of the user interface. In other words, widgets are configurations (instructions) for different parts of the UI.**

- Placing the widgets together creates the **widget tree.** For example, say an architect draws a blueprint of a house; all of the objects like walls, windows, and doors in the house are the widgets, and all of them work together to create the house or, in this case, the application.

Category of Widgets

- There are mainly 14 categories in which the flutter widgets are divided.

They are mainly separated on the basis of the functionality they provide in the flutter app.

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 which are absolutely necessary for the development of any flutter application.
6. Cupertino: These are the ios designed widgets.
7. Input: This set of widgets provide input functionality in a flutter application.

Category of Widgets

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 scroll ability 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.

Stateful versus stateless widgets

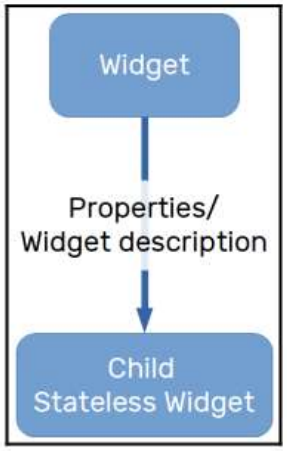
- We have seen so far that widgets play an important role in Flutter application development. They are the pieces that form the UI; they are the code representation of what is visible to the user.

- That's why Flutter provides us with two types of widgets: stateless and stateful.

- The big difference between these is in the way the widget is built. It's the developer's responsibility to choose which kind of widget to use in each situation to compose the UI in order to make the most of the power in the widget rendering layer of Flutter.

Stateless widgets

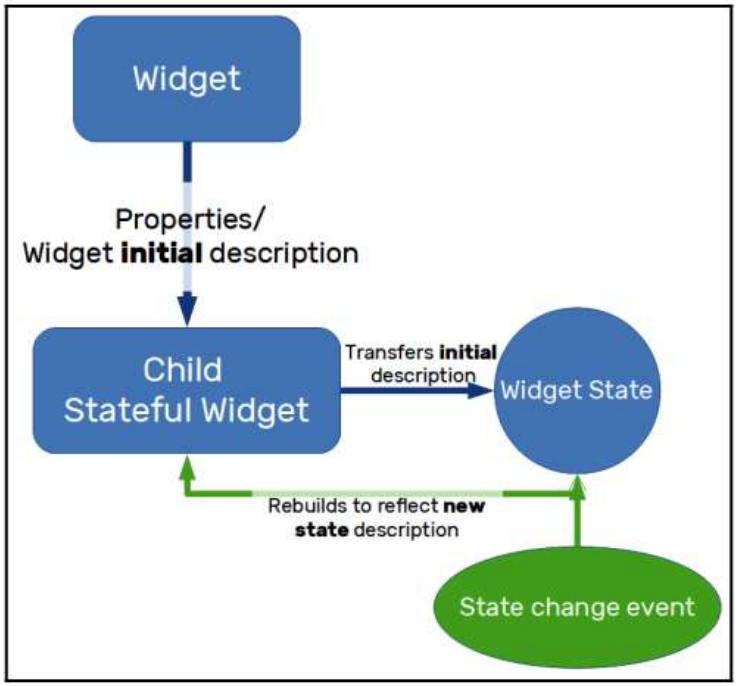
- A typical UI will be composed of many widgets, and some of them will never change their properties after being instantiated. They do not have a state; that is, they do not change by themselves through some internal action or behavior. Instead, they are changed by external events on parent widgets in the widgets tree. So, it's safe to say that stateless widgets give control of how they are built to some parent widget in the tree. The following is a representation of a stateless widget:



The child widget will receive its description from the parent widget and will not change it by itself. In terms of code, this means that stateless widgets have only final properties defined during construction, and that's the only thing that needs to be built on the device screen.

Stateful widgets

- Unlike stateless widgets, which receive a description from their parents that persist during the widgets' lifetime, stateful widgets are meant to change their descriptions dynamically during their lifetimes. By definition, stateful widgets are also immutable, but they have a company State class that represents the current state of the widget. It is shown in the NEXT diagram:



Example: The Layout Tree of basic app screen:

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

void main() => runApp(Test1());

class Test1 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: Scaffold(backgroundColor: Colors.white,
    ├─ appBar: AppBar(backgroundColor: Colors.blue,
    │  └─ title: Text("بسم الله الرحمن الرحيم"),),  // AppBar

    ├─ body: Container(child: Center(child: Text("Welcome to ITMC323 Course"),),),),),);}}  // Scaffold,
```

Example: The Layout Tree of basic app screen:

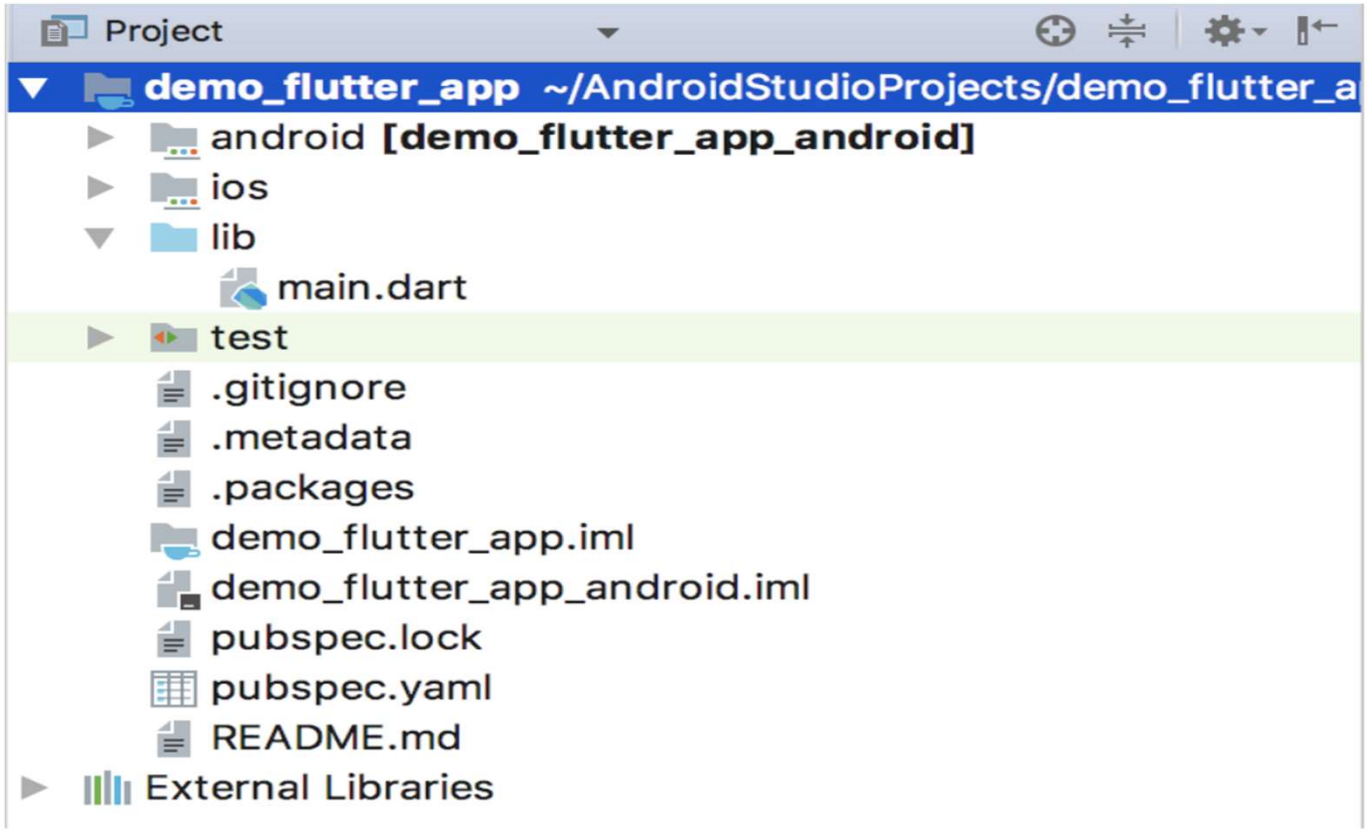
- **Description of the Widgets Used:**

- Scaffold – Implements the basic material design visual layout structure.

- AppBar – To create a bar at the top of the screen.
- Text To write anything on the screen.
- Container – To contain any widget.
- Center – To provide center alignment to other widgets.

Exploring 1st Flutter Project

We'll come back to the code in a while, let's understand the files in the project before that.



Project Directories

- To make a basic app, you only need to focus on the **lib directory** and the **pubspec.yaml** file.

- The **“lib” directory** holds all the **main dart code** used to run your app where as the “pubspec.yaml” file contains all of the packages you’ve imported. (For Android Developers: This is equivalent to your gradle files where you add in dependencies)