Mobile Application
State and gesture

Flutter State Management

The widget can be classified into two categories, one is a **Stateless widget,** and another is a **Stateful widget.** The Stateless widget does not have any internal state. It means once it is built, we cannot change or modify it until they are initialized again. On the other hand, a Stateful widget is dynamic and has a state. It means we can modify it easily throughout its lifecycle without reinitialized it again.

Stateful Widget

- A stateful Widget means a widget that has a mutable state. The state is information that can be read synchronously when the widget is built and might change during a widget's lifetime.

Creating a stateful widget

- A stateful widget is implemented by two classes: a subclass of **StatefulWidget** and a subclass of **State**.
- The **state** class contains the widget's mutable state and the widget's **build()** method.
- When the widget's state changes, the state object calls **setState()**, telling the framework to redraw the widget.

StatefulWidget

```
class Myclass extends StatefulWidget {

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

class _MyclassState extends State<Myclass> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

What is State?

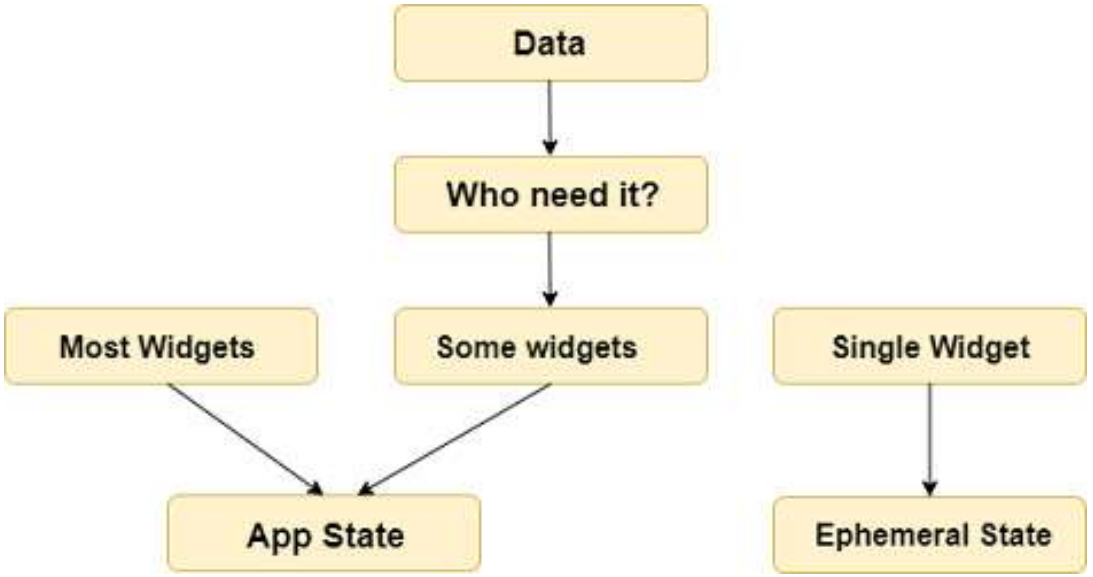
A state is information that can be **read** when the widget is built and might **change or modified** over a lifetime of the app. If you want to change your widget, you need to update the state object, which can be done by using the setState() function available for Stateful widgets. The **setState()** function allows us to set the properties of the state **object** that triggers a redraw of the UI.

Types of State

- In Flutter, the state management categorizes into two conceptual types, which are given below:

1.Ephemeral State

2.App State



Ephemeral State

- This state is also known as UI State or local state. It is a type of state which is related to the **specific widget,** or you can say that it is a state that contains in a single widget.

App State

- It is different from the ephemeral state. It is a type of state that we want to **share** across various parts of our app and want to keep between user sessions. Thus, this type of state can be used globally. Sometimes it is also known as application state or shared state.

Example

- in this example we are learning first counter app in flutter.

Tools:

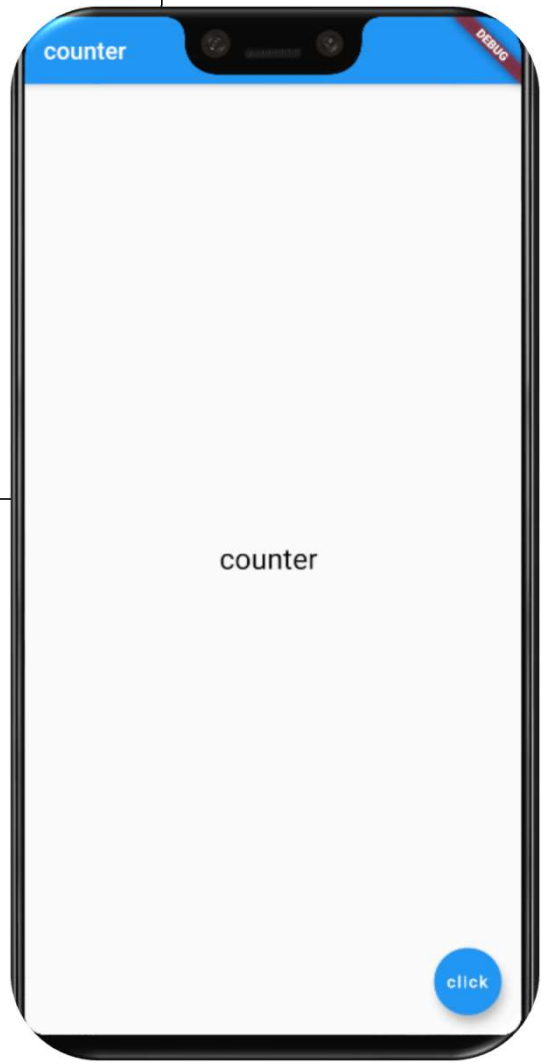
- Main method
- runApp
- MaterialApp → home
- Scaffold → appbar and body
- Floating action button
- setState()

setState:

- setState is one of state management pattern in flutter.
- Calls setState() to update the UI. If remove the setState can not data refresh your screen.

```
import 'package:flutter/material.dart';
void main(){
  runApp(Myapp());
}
class Myapp extends StatefulWidget {
  @override
  _MyappState createState() => _MyappState();
}
class _MyappState extends State<Myapp> {
 int count=0; //define a variable
@override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text(“ App counter"),
        ),
        body: Center(
          child: Text("counter $count ",style:
TextStyle(
            fontSize: 24,color: Colors.black,
          ),)
        ),
```

```
floatingActionButton: FloatingActionButton(
  onPressed: (){
setState (((){
count+=1;
});
},
child: Text("click"),
),
      ),
    );
  }
}
```



InkWell

- The InkWell widget in Flutter is used to make interactive elements respond to touch gestures. It provides a visual splash or highlight effect when tapped

Gestures

- Gestures are an interesting feature in Flutter that allows us to interact with the mobile app (or any touch-based device). Generally, gestures define any physical action or movement of a user in the intention of specific control of the mobile device. Some of the examples of gestures are:
- When the mobile screen is locked, you slide your finger across the screen to unlock it.
- Tapping a button on your mobile screen, and
- Tapping and holding an app icon on a touch-based device to drag it across screens.

Gesture Detector

- The **GestureDetector** is a non-visual widget primarily used for detecting the user's gesture. To identify a gesture targeted on a widget, the widget can be placed inside GestureDetector widget.

Tap – Touching the surface of the device with fingertip for a short period and then releasing the fingertip.

```
home: Scaffold(
      appBar: AppBar(
        title: Text("counter number"),
      ),
      body: Center(
        child:
          Container(
    child:GestureDetector(
      onTap: (){
        print("hello");
      },
      child: Center(
        child: Container(
          child: Text("click me",style: TextStyle(
            color: Colors.white,
    ),),  ), ), ),
              height: 60,
              width: 100,
              color: Colors.black,
            ),  ), ),
  ); }}
```

Double Tap: It is similar to a Tap gesture, but you need to tapping twice in a short time. This gesture contains the following events:

- **onDoubleTap**

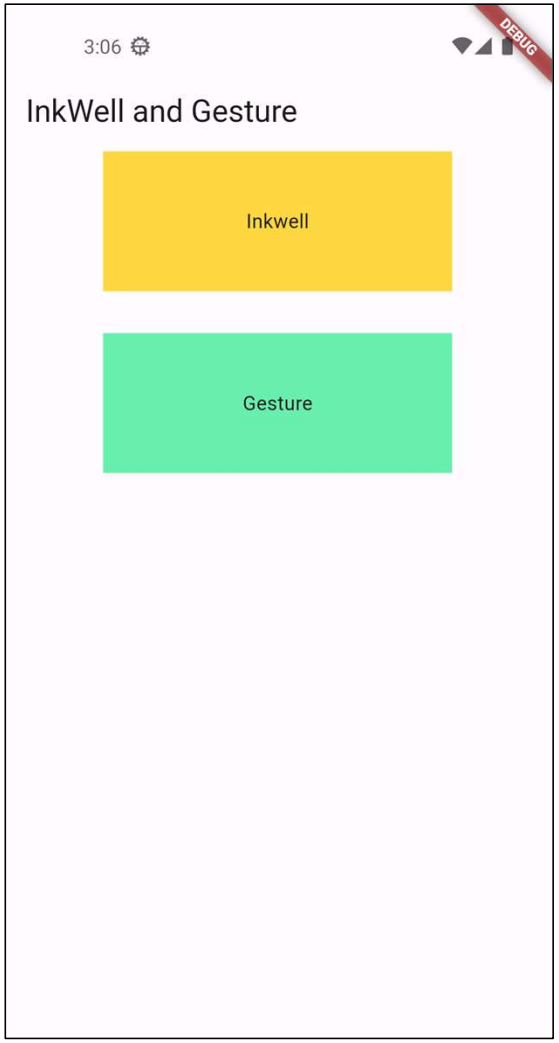
```
home: Scaffold(
      appBar: AppBar(
        title: Text("counter number"),
      ),
      body: Center(
        child:
          Container(
    child:GestureDetector(
      onDoubleTap: (){
        print("hello");
      },
      child: Center(
        child: Container(
          child: Text("click me",style: TextStyle(
            color: Colors.white,
    ),),  ), ), ),
              height: 60,
              width: 100,
              color: Colors.black,
            ),  ), ),
  ); }}
```

Homework

- Pinch – Pinching the surface of the device using two fingers.
- Spread/Zoom – Opposite of pinching.
- Drag

```
Scaffold(
appBar: AppBar(
title: const Text("InkWell and
Gesture"),),
body: Center(
child: Column(
children: [
InkWell(
onTap: () {    print("ink");  },
child: Container(
height: 100,
width: 250,
color: Colors.amberAccent,
alignment: Alignment.center,
child: const Text("Inkwell"),
              ),      ),
```

```
 const SizedBox(
  height: 30,  ),
GestureDetector(
onTap: () {  print("gesture"); },
child: Container(
height: 100,
width: 250,
color: Colors.greenAccent,
alignment: Alignment.center,
child: const Text("Gesture"),
              ),
            ),
          ],
        ),
      ),
    );
```



?