Stateful Widget

State

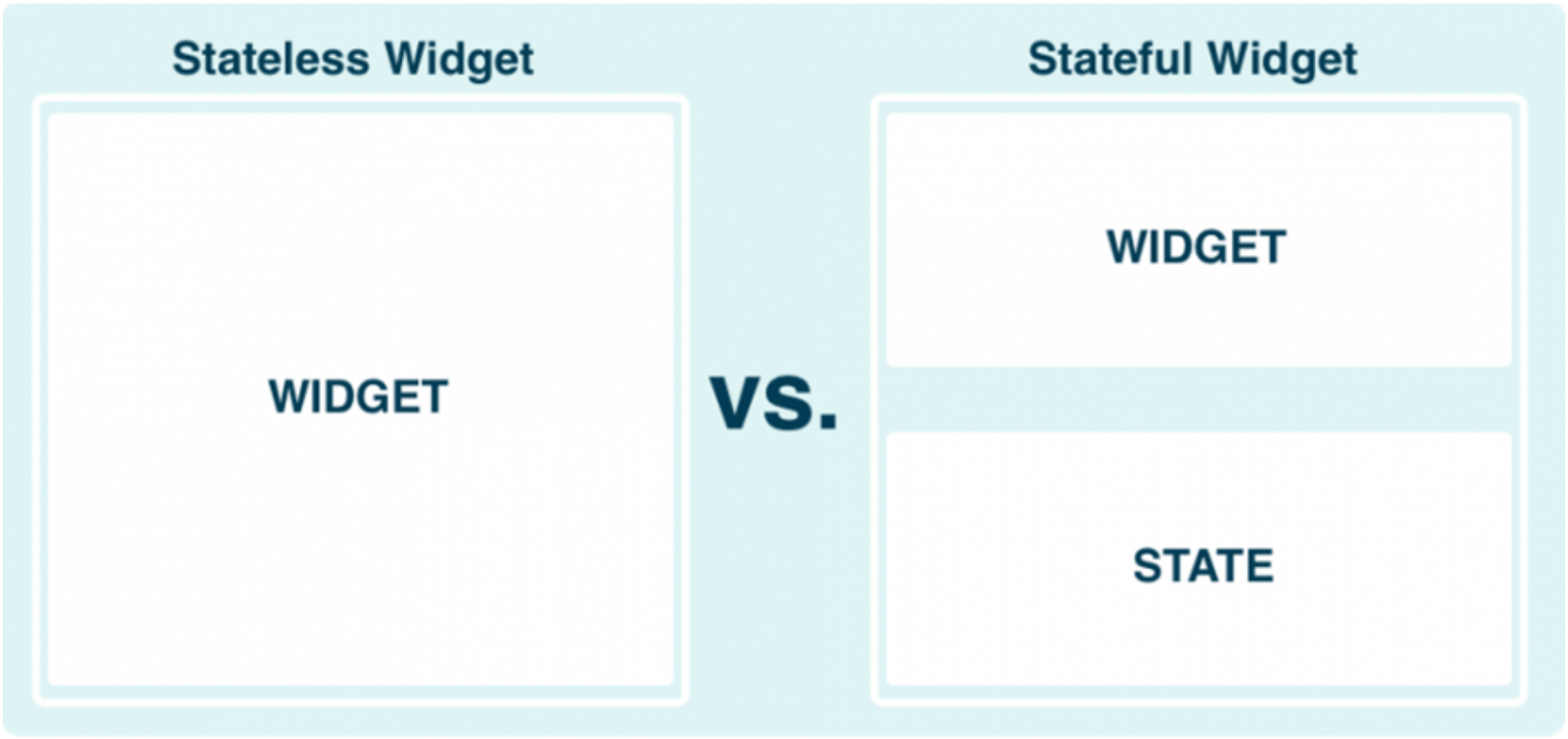
- First Gear
- Pickup Increase
- Second Gear
- Pickup Increase
- Third Gear
- Forth Gear
- Gear Shift Down
- Pickup Decrease

State

What is state:

- Anything that exists in the memory of the app while the app is running.
- When state values change, view update automatically

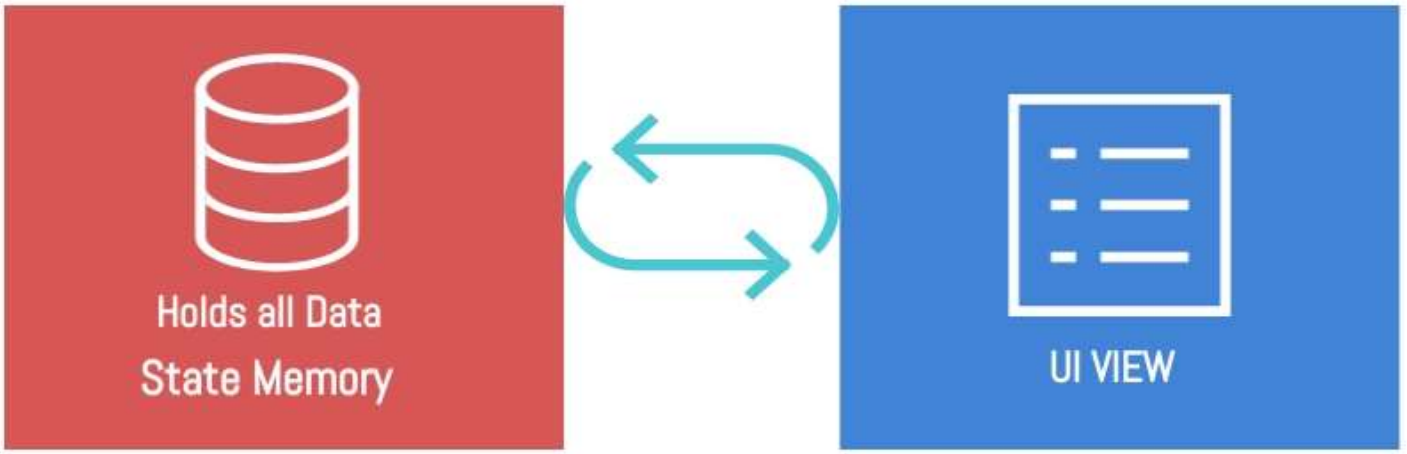
Stateless & Stateful

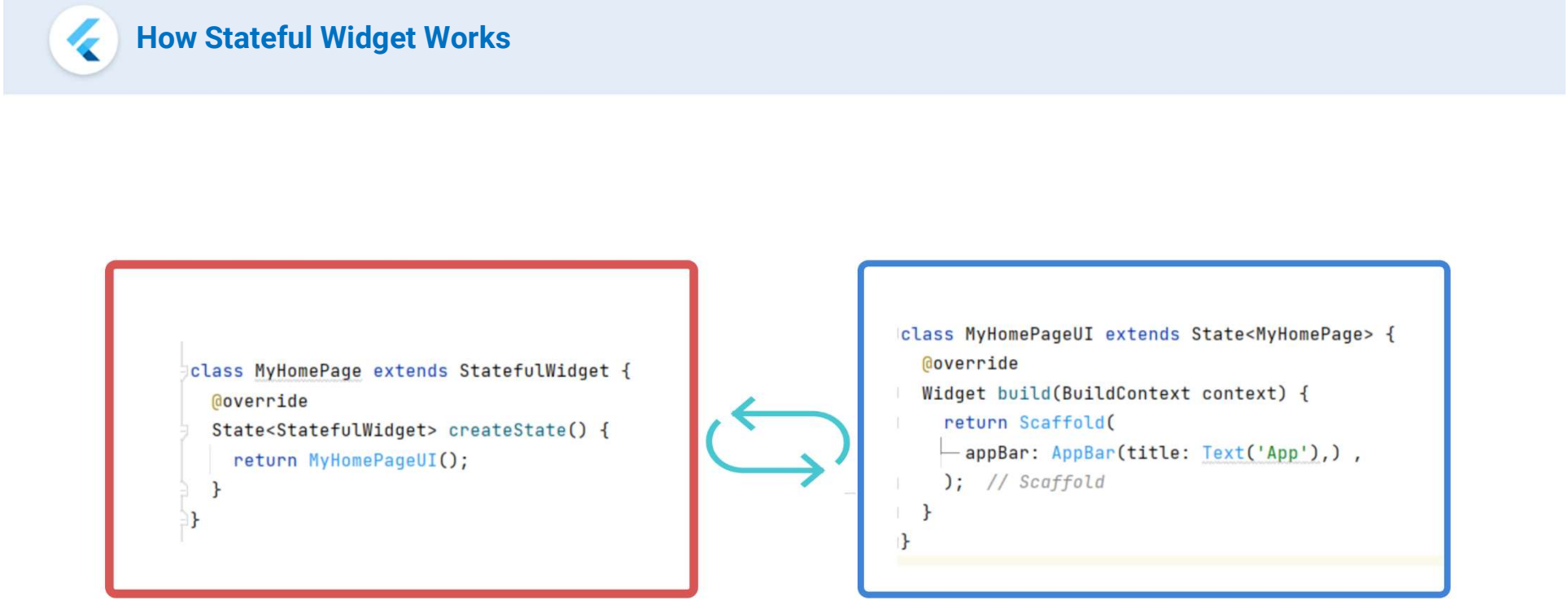


Stateless & Stateful

Stateless	Statefull
Not update at runtime	Update at runtime
Update when initiate	Update when initiate
Can't Change UI Dynamically	Can change UI dynamically
No information caching	Can cache information
Not suitable for API calling	Suitable for API call
Not suitable for complex UI	Suitable for complex UI

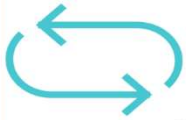
How Stateful Widget Works




How Stateful Widget Works
class MyHomePage extends StatefulWidget {
@override
State<StatefulWidget> createState() {
return MyHomePageUI();
}
}
class MyHomePageUI extends State<MyHomePage> {
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(title: Text('App'),) ,
); // Scaffold
}
}

Lets Create Our First Stateful Widget

```dart
class MyHomePage extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return MyHomePageUI();
  }
}
```



```dart
class MyHomePageUI extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('App'),) ,
    );  // Scaffold
  }
}
```

State Lifecycle methods

- The Stateful widget is mutable
- It can be change multiple times within its lifetime.
- So we need to understand it's lifecycle

State Lifecycle methods

createState()

- This method creates a State object. This object is where all the mutable state for that widget is held.
- This step is not marked as a real step in the lifecycle, but it is important to know what is going on in the background.

State Lifecycle methods

initState()

- Automatically called only once, when the state object is created for the first time.
- Use this method to manage HTTP requests and subscribe to streams or any other object that could change the data on this widget.

State Lifecycle methods

didChangeDependencies()

- Framework will call this method immediately after the **initState**().
- The build method is always called after this method, so this method is rarely needed

State Lifecycle methods

build()

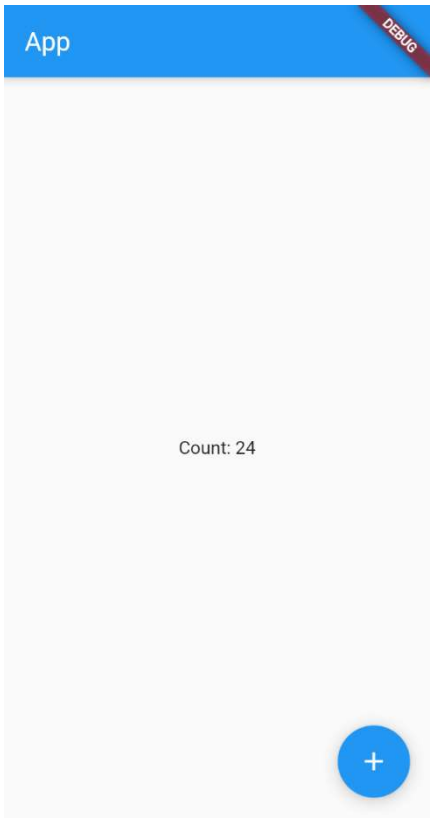
- it will be called many times during the lifecycle, but the first time is after the **didChangeDependencies**() method.
- Whenever the widget that belongs to the state is updated, the framework will always execute this method

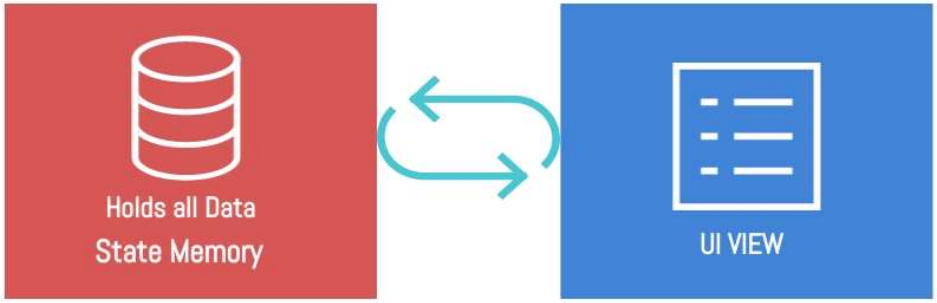
State Lifecycle methods

setState()

- The setState() method notifies the framework that the internal state of the current object is changed, now it's time to update the view

Lets Create A Counter App





Lets Create Sum Calculator