Lifecycle of Stateful Widget

The lifecycle of a stateful Widget explains the calling hierarchy of functions and changes in the state of the widget.
Everything in the flutter is a widget.
Basically, the Stateful and Stateless are the types of widgets in the flutter.

Stateful Widget

Stateful widgets are widgets that can be changed dynamically on user interaction or input.
It is drawn multiple times during its lifetime.

Life Cycle Methods Of StateFul Widgets

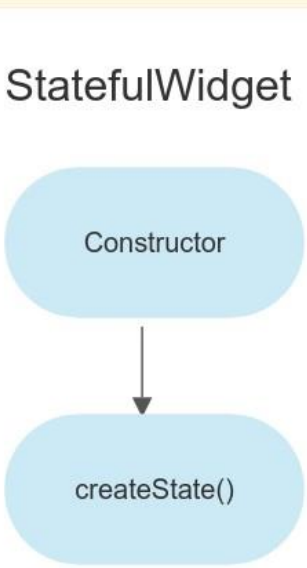
When we create a stateful widget we have to extend StatefulWidget class and another class that extends the State class.

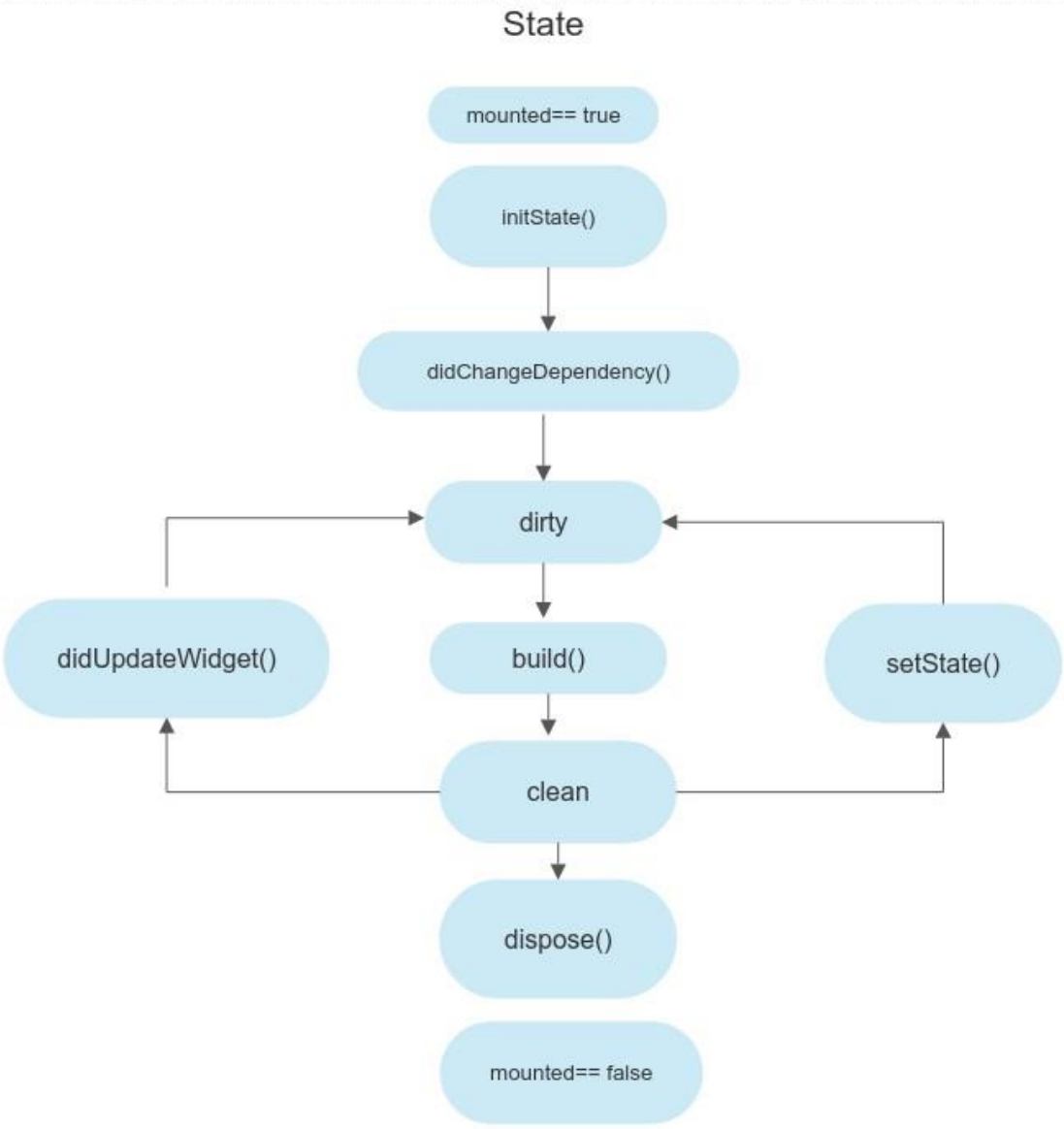
Create State() :

It is the required method of statefulWidget which must be overridden.

```
1 class LifecycleSample extends StatefulWidget {
```

```
2   @override
3   _MyPageState createState() => LifecycleSampleState();
4 }
```


StatefulWidget
Constructor
createState()



Mounted :

This property set to true once the **state** object is created.

This property is useful to check whether a state is in the widget tree.

We can avoid errors by checking this property while rebuilding the widget or calling the setState function.

InitState():

It is the first calling method of the **state**.

It is called only once when we create a widget.

We can use this method for initializing purposes.

```
1 @override
2 void initState() {
3 //variable can be intialized here.
4   super.initState();
5
6 }
```

DidChangeDependencies():

This method is called immediately after initState method.

This method is rarely used because the build method is always called after this method.

It is also called when a dependency of this State object changes.

```
1  @override
2   void didChangeDependencies() {
3     super.didChangeDependencies();
4   }
```

Build():

This is a required method and is called many times during the lifecycle.

This method is also called

after `didChangeDependencies()` method.

```
1  @override
2   Widget build(BuildContext context) {
3    return Text(" Sample app");
```

```
4  }
```

DidUpdateWidget() :

This method is called at the time of widget rebuild and the parent widget changes its configuration.

It gives an old widget as an argument that can be used for widget comparison.

Build method will call just after it.

```
1  @override
2  void didUpdateWidget(covariant _sampleLifecycle oldWidget) {
3    super.didUpdateWidget(oldWidget);
4  }
```

SetState() :

this method is called by developers when there is a requirement to rebuild the widget.

Framework called this function itself.

setState() notify to the framework that the widget should be rebuilt as data has been changed.

```
1 setState(() {
2
3 });
```

Dispose() :

This method is called when the state object is removed from the Widget tree.

this method can be used for cleaning purposes like removing listener etc.

```
1  @override
2   void dispose() {
3     super.dispose();
4   }
```

Mounted (False):

After dispose() this property is set to false.

If setState is called when this property is false will throw to error.