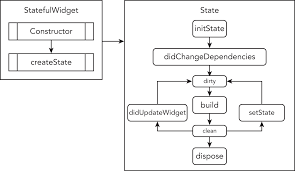
**Widget Lifecycle Methods:**

The life cycle is based on the state and how it changes. A stateful widget has a state so we can explain the life cycle of flutter based on it.



**Stage of the life cycle:**

**• createState():** When we create a stateful widget, our framework calls a createState() method and it must be overridden

class MyPage extends StatefulWidget{

@override

\_MyPageState **createState**() => \_MyScreenState();

}

**• initState():** is the first method that is used while creating a stateful class, here we can initialize variables, data, properties, etc. for any widget.

@override

void **initState**() {

a= 0;

super.**initState**();

}

• **didChangeDependencies**()

@override

void didChangeDependencies() {

super.didChangeDependencies()

}

**• build()** **:** The build method is used each time the widget is rebuilt. This can happen either after calling initState, didChangeDependencies, didUpdateWidget, or when the state is changed via a call to setState

@override

Widget build(BuildContext context, MyButtonState state) {

return Container(color:Colors.red);

}

**• didUpdateWidget():** This method is called whenever the widget configuration changes. A typical case is when a parent passes some variable to the children() widget via the constructor.

@override

void didUpdateWidget(MyHomePage oldWidget) {

super.didUpdateWidget(oldWidget)

}

**• setState() :** method is called often from the Flutter framework itself and from the developer

void updateProfile(String name) {

setState(() => this.name = name);

}

**• deactivate() :** It is used when the state is removed from the tree but before the current frame change can be re-inserted into another part of the tree

@override

void deactivate() {

super.deactivate();

}

**dispose():** We use this method when we remove permanently like should release resource created by an object like stop animation

@override dispose() {

animationController.dispose();

super.dispose();

}