The Stateful life cycle of Stateful widget

Stateful widgets are used when we want objects to be updated on the screen , for example when a user presses a button to trigger an action , these widgets are mutable they can change or updated theie appearance over time..

The main stages in the lifecycle of a stateful widgets:

createSate() is the first method that's called when a stateful widget is inserted into a widget tree , it creates an instance of the state class associated with the widget.

```
_MyHomePageState createState() => new _MyHomePageState();
```

mounted it is a bool value , that turns true when the buildContext is assigned to the widget.

initState () is called just before the widget gets built, this is where you can initialize data or perform setup operations, it runs only once during the widget's lifecycle.

```
@override
initState() {
  super.initState();}
```

didChangeDependencies()is called just after the initstate() and it can be called multiple times.

build() is responsible for returning the widget's user interface , this method rebuilds when setState is called.

didUpdateWidget()is called if the parent widget changes and triggers a rebuild of the

stateful widget

```
@override
void didUpdateWidget(Widget oldWidget) {
  if (oldWidget.importantProperty != widget.importantProperty) {
    _init();
  }
}
```

setState() is used to signal that widget internal state has changed and that a rebuild is needed, when setState() is called it schedules a call to the build method and marks the widget as needing to be rebuilt.

```
void updateProfile(String name) {
 setState(() => this.name = name);
}
```

deactivate() is called when the widget is popped but it might be reinserted before the current frame change is finished.

dispose() is called when the widget is removed permanetly from the widget tree.