Flutter life cycle

- Flutter has its own unique lifecycle that manages the state of the widget tree and handles events such as creation, updates, and destruction of widgets. The lifecycle of a Flutter app is as follows

`main()` method: This is the entry point of a Flutter app. It is where the app is initialized, and it typically calls the `runApp()` function.

Stages In Life Cycle Of Flutter App

CreateState()

This method is called when we create a new Stateful Widget.It is a mandatory method . It will return an instance of a State associated with it.

```
class Home extends StatefulWidget {
  @override
  HomeState<StatefulWidget> createState() => Home();
}
```

InitState()

This is the method which is called when the Widget is created for the first time and it is called exactly once for each State object.

If we define or add some code in initState() method then this code will execute first even before the widgets are being built.

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

DidChangeDependencies()

This method is called immediately after the initState() method on the first time the widget is built.

```
@override
void didChangeDependencies() {

}
```

Build()

This method is the most important method as the rendering of all the widgets depends on it.

It is called every time when we need to render the UI Widgets on the screen.

```
@override
Widget build(BuildContext context) {
//add your widgets
}
```

DidUpdateWidget()

This method is used when there is some change in the configuration by Parent widget.

It is basically called every time we hot reload the app for viewing the updates made to the widget.

```
@protected
void didUpdateWidget(Home oldWidget) {
  super.didUpdateWidget(oldWidget);
}
```

SetState()

The setState() method informs the framework that the internal state of this object has changed in a way that might impact the UI which causes the framework to schedule a build for this State of object.

It is an error to call this method after the framework calls dispose.

```
setState(() {

});
```

Deactivate()

This method is called when the State is removed from the tree, but this method can be also be re-inserted into the tree in some other part.

```
@override
void deactivate(){
  super.deactivate();
}
```

Dispose()

This method is basically the opposite of initState() method and is also very important.

It called when the object and its State needs to be removed from the Widget Tree permanently and will never build again.

```
@override
void dispose(){
   super.dispose();
}
```