**1. Difference Between Stateless and Stateful Widgets**

In Flutter, everything is a widget, and widgets can be classified into **Stateless** and **Stateful** widgets based on their ability to maintain state.

**Stateless Widgets**

A **Stateless widget** does not change once it is built. It remains the same throughout its lifecycle and does not store any mutable state.

**Example of Stateless Widget**

import 'package:flutter/material.dart';

class MyStatelessWidget extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text("Stateless Widget")),

body: Center(child: Text("This is a Stateless Widget")),

);

}

}

void main() {

runApp(MaterialApp(home: MyStatelessWidget()));

}

**Key Features of Stateless Widgets:**

* Cannot change dynamically.
* Does not store user interactions or update data.
* Rebuilt only when necessary.

**Stateful Widgets**

A **Stateful widget** can change dynamically as it stores and manages state. When the state changes, the widget rebuilds to reflect the changes.

**Example of Stateful Widget**

import 'package:flutter/material.dart';

class MyStatefulWidget extends StatefulWidget {

@override

\_MyStatefulWidgetState createState() => \_MyStatefulWidgetState();

}

class \_MyStatefulWidgetState extends State<MyStatefulWidget> {

int counter = 0;

void incrementCounter() {

setState(() {

counter++;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text("Stateful Widget")),

body: Center(child: Text("Counter: $counter")),

floatingActionButton: FloatingActionButton(

onPressed: incrementCounter,

child: Icon(Icons.add),

),

);

}

}

void main() {

runApp(MaterialApp(home: MyStatefulWidget()));

}

**Key Features of Stateful Widgets:**

* Can change dynamically based on user interactions.
* Stores state and updates UI accordingly.
* Uses setState() to notify the framework of changes.

**2. Widget Lifecycle and State Management in Stateful Widgets**

Stateful widgets have a **lifecycle** that manages the widget from creation to destruction.

**Lifecycle of a Stateful Widget**

1. **createState()** → Creates the state object.
2. **initState()** → Called once when the widget is inserted into the tree.
3. **didChangeDependencies()** → Called when inherited widgets change.
4. **build()** → Builds the UI every time the state changes.
5. **setState()** → Updates the state and rebuilds the UI.
6. **deactivate()** → Called before a widget is removed from the tree.
7. **dispose()** → Called when the widget is permanently removed (cleanup).

**Example of Lifecycle Methods in a Stateful Widget**

import 'package:flutter/material.dart';

class LifecycleDemo extends StatefulWidget {

@override

\_LifecycleDemoState createState() => \_LifecycleDemoState();

}

class \_LifecycleDemoState extends State<LifecycleDemo> {

@override

void initState() {

super.initState();

print("initState() called");

}

@override

Widget build(BuildContext context) {

print("build() called");

return Scaffold(

appBar: AppBar(title: Text("Widget Lifecycle")),

body: Center(child: Text("Check console for lifecycle methods")),

);

}

@override

void dispose() {

print("dispose() called");

super.dispose();

}

}

void main() {

runApp(MaterialApp(home: LifecycleDemo()));

}

**3. Five Common Flutter Layout Widgets**

Flutter provides various layout widgets for designing user interfaces. Here are five commonly used layout widgets:

**1. Container**

* Used for holding a child widget.
* Can be styled with padding, margin, color, border, and decoration.

**Example**

Container(

width: 200,

height: 100,

color: Colors.blue,

child: Center(child: Text("Hello Flutter!")),

)

**2. Column**

* Arranges widgets vertically.
* Useful for creating vertical layouts.

**Example**

Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text("Hello"),

Text("Flutter"),

],

)

**3. Row**

* Arranges widgets horizontally.
* Similar to Column but in a horizontal direction.

**Example**

Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

Icon(Icons.star),

Icon(Icons.favorite),

Icon(Icons.thumb\_up),

],

)

**4. Stack**

* Overlaps multiple widgets on top of each other.
* Useful for placing elements freely within a defined area.

**Example**

Stack(

children: [

Container(width: 200, height: 200, color: Colors.blue),

Positioned(top: 50, left: 50, child: Icon(Icons.star, size: 50, color: Colors.white)),

],

)

**5. ListView**

* Displays a scrollable list of widgets.
* Used when the number of items is dynamic.

**Example**

ListView(

children: [

ListTile(title: Text("Item 1")),

ListTile(title: Text("Item 2")),

ListTile(title: Text("Item 3")),

],

)