Stateless vs Stateful

Lec 5 search Topic

Nada Khaled El-Torogoman
ERAA SOFT Flutter 127

Stateless vs Stateful in Flutter

◆ Stateless Widget

"I don't change. What you see is what you get."

- These widgets **never change their internal state** after they're built.
- Think of them like pure UI elements that depend **only on the inputs (constructor data)**.
- Great for simple UI components: headers, buttons, icons, etc.

Example:

```
class MyTextWidget extends StatelessWidget {
  final String text;

  const MyTextWidget({super.key, required this.text});

  @override
  Widget build(BuildContext context) {
    return Text(text);
  }
}

```

◆ Stateful Widget

"I evolve. I can react to taps, time, or data updates."

- These widgets **can rebuild themselves** when the internal state changes.
- You get access to a **State** object where you store and update data using **setState()**.
- Perfect for forms, animations, toggles, tab views, and anything interactive.

Example:

```
class CounterWidget extends StatefulWidget {
  const CounterWidget({super.key});

  @override
  State<CounterWidget> createState() => _CounterWidgetState();
}

class _CounterWidgetState extends State<CounterWidget> {
  int count = 0;

  void increment() {
    setState(() {
      count++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Text('Count: $count'),
        ElevatedButton(onPressed: increment, child: const Text('Increment')),
      ],
    );
  }
}

```

🧠 TL;DR (aka Flutter cheat sheet):

Feature	**Stateless Widget**	**Stateful Widget**
Holds state?	❌ Nope	✅ Yes
Rerenders on change?	❌ Only on parent rebuild	✅ On setState()
Use when?	🎯 UI is fixed/static	✨ UI reacts/changes
Performance	🔥 Fast (less overhead)	🧠 Slightly heavier

Got a UI piece that just listens and shows? Go Stateless.

If it talks back, changes, or dances, go Stateful.