Flutter Custom Widgets: Reusable Components for Scalable Apps

Subtitle: Learn how to create and use custom widgets to simplify Flutter app development.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Custom widgets allow you to **reuse UI components** and maintain **clean, scalable code**. Flutter's widget system makes it easy to create both **stateless and stateful reusable components**.

Step 1: Stateless Custom Widget

```
class CustomButton extends StatelessWidget {
  final String text;
  final VoidCallback onPressed;

  const CustomButton({required this.text, required this.onPressed, Key? key}) :

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: onPressed,
      child: Text(text),
    );
  }
}
```

Exercise: Use `CustomButton` in your app with different labels.

Step 2: Stateful Custom Widget

```
class CounterWidget extends StatefulWidget {
  const CounterWidget({Key? key}) : super(key: key);

  @override
  State<CounterWidget> createState() =>  CounterWidgetState();
```

```
}

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

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Text('Count: $_count'),
        ElevatedButton(
          onPressed: () => setState(() => _count++),
          child: Text('Increment'),
        ),
      ],
    );
  }
}
```

Exercise: Add multiple instances of `CounterWidget` on the same screen.

Step 3: Parameterized Widgets

- Pass **colors, icons, and text** to customize behavior and look:

```
class ProfileCard extends StatelessWidget {
  final String name;
  final String imageUrl;

  const ProfileCard({required this.name, required this.imageUrl, Key? key}) : su

  @override
  Widget build(BuildContext context) {
    return Card(
      child: ListTile(
        leading: Image.network(imageUrl),
        title: Text(name),
      ),
    );
  }
}
```

Exercise: Create a list of users using `ProfileCard`.

Step 4: Composition & Nesting

- Build complex UI by **combining multiple custom widgets**:

```
Column(
  children: [
    CustomButton(text: "Click Me", onPressed: () {}),
    CounterWidget(),
    ProfileCard(name: "Hafsa", imageUrl: "https://example.com/avatar.png"),
  ],
)
```

Exercise: Design a small dashboard using at least 3 custom widgets.

Step 5: Mini Project

Build a **Reusable Component Library**:

- Buttons with different styles
- Card widgets for displaying data
- A counter widget for interactive UI
- Combine them on a single screen to form a small dashboard

Key Takeaways

- Custom widgets make your code **modular and reusable**
- Stateless widgets are simple and lightweight
- Stateful widgets handle dynamic content
- Parameterization and composition improve scalability
- Mastering custom widgets is essential for **maintainable Flutter apps**

Ignoring custom widgets leads to **messy, repetitive code**, making your app hard to maintain and scale.

Visit **haas.dev** for more Flutter guides, tutorials, and complete project resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app
