Flutter Custom Widgets: Building Reusable UI Components

Subtitle: Learn how to create modular and reusable widgets to make your Flutter code cleaner and scalable.

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

Introduction

Custom widgets help you **restructure your UI**, reduce code repetition, and make your app **easier to maintain**. This guide teaches how to create and use reusable widgets.

Step 1: Create a Stateless Custom Widget

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

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

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

Exercise: Use `CustomButton` in your main screen with different labels.

Step 2: Create a Stateful Custom Widget

```
class ToggleWidget extends StatefulWidget {
  @override
  _ToggleWidgetState createState() => _ToggleWidgetState();
}
```

```
class _ToggleWidgetState extends State<ToggleWidget> {
  bool isOn = false;

  @override
  Widget build(BuildContext context) {
    return Switch(
      value: isOn,
      onChanged: (value) {
        setState(() {
          isOn = value;
        });
      },
    );
  }
}
```

Exercise: Add multiple `ToggleWidget` instances with independent states.

Step 3: Widget with Custom Parameters

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

  const ProfileCard({required this.name, required this.role, Key? key}) : super(

  @override
  Widget build(BuildContext context) {
    return Card(
      child: ListTile(
        title: Text(name),
        subtitle: Text(role),
      ),
    );
  }
}
```

Exercise: Display a list of profile cards using a list of names and roles.

Step 4: Compose Widgets Together

```
Column(
```

```
  children: [
    ProfileCard(name: 'Hafsa', role: 'Developer'),
    ProfileCard(name: 'Ali', role: 'Designer'),
    CustomButton(label: 'Click Me', onPressed: () {}),
  ],
)
```

Exercise: Combine multiple custom widgets into one screen to build a dashboard.

Step 5: Mini Project

Build a **Flutter Dashboard**:

- Reusable `ProfileCard` and `CustomButton`
- Combine multiple widgets in a `Column` or `GridView`
- Update one widget and see it reflect wherever used

Key Takeaways

- Custom widgets improve **code reusability and readability**
- Stateless widgets are **lightweight**, Stateful widgets handle **dynamic UI**
- Passing parameters makes widgets **flexible for multiple uses**
- Modular design is essential for **scalable and maintainable apps**

Skipping custom widgets leads to **repetitive, messy code**, which becomes hard to manage as your app grows.

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

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