Flutter Widgets & Layouts: Complete Beginner Guide

Subtitle: Learn how to build responsive and visually structured UIs using Flutter widgets and layout system.

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

Introduction

Flutter UI is built entirely with widgets. Understanding how widgets work and how to structure layouts is essential for building real apps. This guide covers **core widgets, layout systems, and best practices** to design clean and responsive interfaces.

Step 1: Understanding Widgets

In Flutter, everything is a widget:

- **Text** → Displays text
- **Container** → Box model (padding, margin, color)
- **Row & Column** → Layout horizontally/vertically
- **Image** → Displays images
- **Icon** → Shows icons

Example:

```
Column(
  children: [
    Text("Hello Flutter"),
    Icon(Icons.star),
  ],
)
```

Exercise: Create a Column with text and an icon below it.

Step 2: Container Widget

`Container` is one of the most used widgets:

- Add **padding, margin, color, border**
- Control size and alignment

Example:

```
Container(
  padding: EdgeInsets.all(16),
  margin: EdgeInsets.all(10),
  color: Colors.blue,
  child: Text("Container Example"),
)
```

Exercise: Create a container with red background and centered white text.

Step 3: Row & Column Layouts

- **Row** → Horizontal layout
- **Column** → Vertical layout

Important properties:

- `mainAxisAlignment`
- `crossAxisAlignment`

Example:

```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: [
    Text("A"),
    Text("B"),
    Text("C"),
  ],
)
```

Exercise: Create a row with three boxes spaced evenly.

Step 4: Expanded & Flexible

Used to control spacing inside Row/Column

Used to control spacing inside Row/Column.

Example:

```
Row(
  children: [
    Expanded(child: Container(color: Colors.red, height: 50)),
    Expanded(child: Container(color: Colors.blue, height: 50)),
  ],
)
```

Exercise: Create two containers with equal width using Expanded.

Step 5: Stack & Positioning

`Stack` allows overlapping widgets.

Example:

```
Stack(
  children: [
    Container(color: Colors.blue, width: 200, height: 200),
    Positioned(
      top: 20,
      left: 20,
      child: Text("Top Text"),
    ),
  ],
)
```

Exercise: Overlay text on top of an image using Stack.

Step 6: ListView & Scrolling

Use `ListView` for scrollable content.

Example:

```
ListView(
  children: [
    Text("Item 1"),
    Text("Item 2"),
    Text("Item 3"),
```

```
  ],
)
```

Exercise: Create a scrollable list of 10 items.

Step 7: Styling & Design Tips

- Use **consistent spacing and colors**
- Prefer **const widgets** for performance
- Avoid deeply nested widget trees
- Use **MediaQuery** for responsive design

Key Takeaways

- Flutter UI is built using **widgets only**.
- Use **Row, Column, and Stack** for layout control.
- `Container` is essential for styling and spacing.
- `Expanded` helps create responsive layouts.
- `ListView` is used for scrollable content.

Mastering widgets and layouts is the foundation for building real Flutter apps.

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

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