

Duhok Polytechnic University

Technical College of Zakho

Department of CIS



# LAB 1: Recalling Basics

---

MOBILE APPLICATION II

# Objectives

---

The Flutter Foundation

Single-Child Architecture

Multi-Child Layouts

Alignment & Positioning

Static UI Composition

# The Flutter Foundation

---

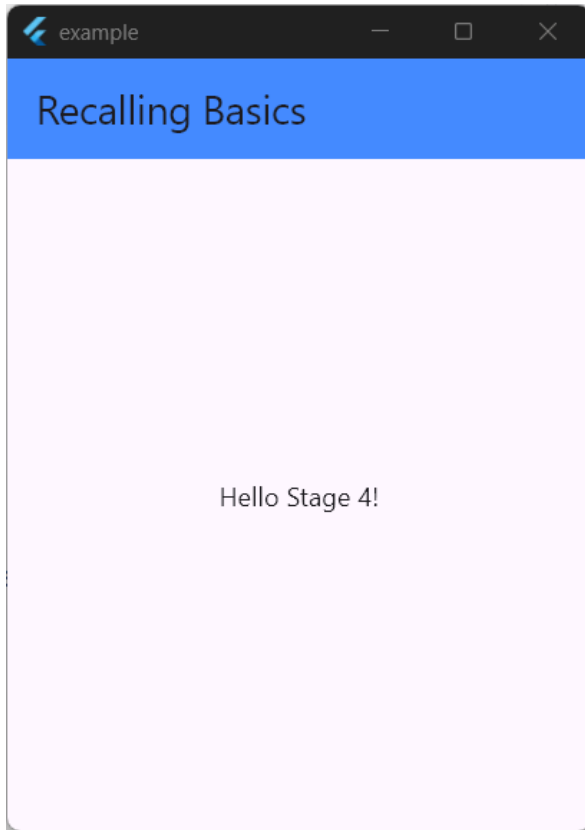
Every standard Flutter application begins with two structural anchors. `MaterialApp` acts as the root configuration of the app, defining the theme, routing, and directional settings. Inside it sits the `Scaffold`, which implements the basic Material Design visual layout structure, providing designated slots for standard UI elements like top bars, body content, and floating actions.

## Key Widgets:

- `MaterialApp`:
  - `home`: The default route or starting screen widget displayed when the app launches normally.
- `Scaffold`:
  - `appBar`: The fixed header slot at the top of the screen.
  - `body`: The primary content area where layout widgets reside.
  - `backgroundColor`: Sets the default canvas color of the entire screen background.

# Foundation Example

---



```
1 MaterialApp(  
2   debugShowCheckedModeBanner: false,  
3   home: Scaffold(  
4     appBar: AppBar(  
5       title: const Text('Recalling Basics'),  
6       backgroundColor: Colors.blueAccent,  
7     ),  
8     body: const Center(child: Text('Hello Stage 4!')),  
9   ),  
10 );
```

# Wrappers & Styling

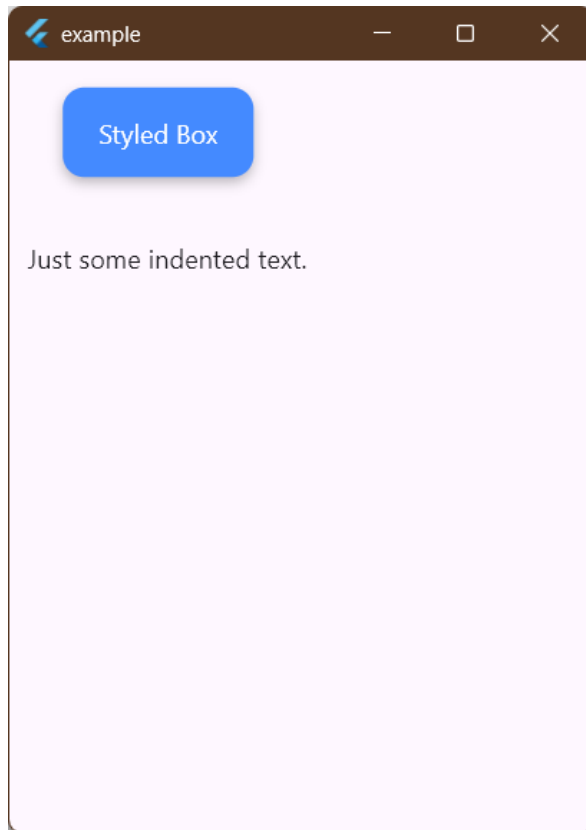
---

Unlike HTML/CSS where styling (margins, padding, shadows) is applied directly as properties to a text or image tag, Flutter uses Wrapper Widgets. Single-child widgets accept exactly one child and are responsible for altering its physical constraints, spacing, or visual presentation.

## Key Attributes & Widgets:

- **Container:** The ultimate "Box Model" widget. It combines padding, margin, sizing, and background styling into one widget.
- **BoxDecoration:** Used exclusively inside a Container's decoration property. It unlocks advanced styling like borderRadius, boxShadow, border, and gradients.
- **Padding:** A lightweight wrapper that only applies inner spacing using EdgeInsets (e.g., .all, .symmetric, .only). Use this instead of a Container if you only need spacing.
- **SizedBox:** Used for two things: forcing a widget to be an exact width and height, or creating an invisible, empty block of space between other widgets.

# Wrappers Example



```
1 Column(  
2   children: [  
3     Container(  
4       margin: const EdgeInsets.all(15),  
5       padding: const EdgeInsets.symmetric(  
6         horizontal: 20,  
7         vertical: 15,  
8       ),  
9       decoration: BoxDecoration(  
10        color: Colors.blueAccent,  
11        borderRadius: BorderRadius.circular(12),  
12        boxShadow: const [  
13          BoxShadow(  
14            color: Colors.black26,  
15            blurRadius: 5,  
16            offset: Offset(0, 3),  
17          ),  
18        ],  
19      ),  
20      child: const Text(  
21        'Styled Box',  
22        style: TextStyle(color: Colors.white),  
23      ),  
24    ),  
25    const SizedBox(height: 20),  
26    const Padding(  
27      padding: EdgeInsets.only(left: 10.0),  
28      child: Text('Just some indented text.'),  
29    ),  
30  ],  
31 )
```

# Row & Column

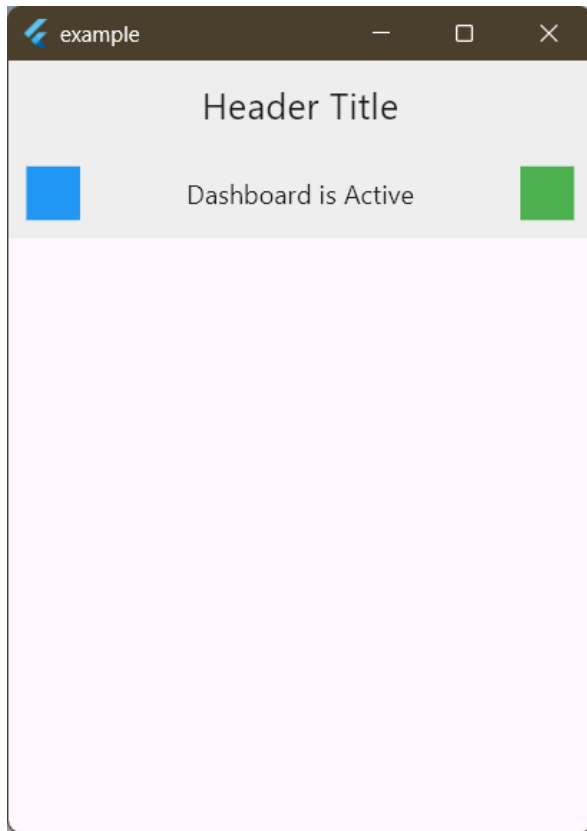
---

When your UI requires multiple elements placed side-by-side or stacked vertically, you graduate from a single child to a children array. The Column (vertical) and Row (horizontal) are the workhorses of Flutter layout. The biggest challenge with these widgets isn't putting items inside them—it's controlling exactly how those items align and distribute themselves across the screen.

## Key Attributes & Axis Logic:

- children: A list of widgets (List<Widget>) rendered in the exact order they are written.
- The Main Axis: The direction the widget naturally flows. (Vertical for Column, Horizontal for Row).
- The Cross Axis: The perpendicular direction. (Horizontal for Column, Vertical for Row).
- mainAxisAlignment: Distributes empty space along the primary axis.
  - Values: .start, .center, .end, .spaceEvenly, .spaceBetween, .spaceAround.
- crossAxisAlignment: Aligns items along the perpendicular axis.
  - Values: .start, .center, .end, .stretch (forces children to fill the cross axis).

# Multi-child Example



```
1 Container(  
2   color: Colors.grey[200],  
3   padding: const EdgeInsets.all(10),  
4   child: Column(  
5     mainAxisAlignment:  
6       MainAxisAlignment.min,  
7     children: [  
8       const Text('Header Title', style: TextStyle(fontSize: 20)),  
9       const SizedBox(height: 20),  
10      Row(  
11        mainAxisAlignment:  
12          MainAxisAlignment.spaceBetween,  
13        crossAxisAlignment:  
14          CrossAxisAlignment.center,  
15        children: [  
16          Container(color: Colors.blue, width: 30, height: 30),  
17          Text('Dashboard is Active'),  
18          Container(color: Colors.green, width: 30, height: 30),  
19        ],  
20      ),  
21    ],  
22  ),  
23 )
```

# Any Questions?

---



# Lab Work

---

