🧪 Flutter Lab 2 — Section 3: Understanding Layouts and Constraints

---

🎯 Objectives

By the end of this lab, students will:

- Understand how Flutter's layout system works through *constraints → size → position*.
- Experiment with layout widgets (`Row`, `Column`, `Stack`, `Container`, `Expanded`, `Align`, etc.).
- Build confidence in controlling widget sizes and positions precisely.

🧩 Concept: The Story of Constraints

"Constraints go down. Sizes go up. Parent sets position."
Flutter layout can't really be understood without knowing this rule, so Flutter developers should learn it early on.

🧠 How Layout Works:

1. **A widget gets constraints** from its parent — a set of min/max width and height.
2. The widget **passes constraints to its children**, possibly modified for each.
3. Each child **chooses a size** within its constraints and reports it back.
4. The parent **positions each child** on the screen.
5. Finally, the widget reports its own size to its parent — still within its constraints.

🪄 Example: A Widget's Conversation

```
Widget: "Hey parent, what are my constraints?"
Parent: "0–300 width, 0–85 height."
Widget: "I have 5px padding, so my children can use 0–290 width, 0–75
height."

Widget → First child: "0–290 width, 0–75 height."
First child → Widget: "I'll take 290x20."

Widget: "OK, now only 55px height left."
Widget → Second child: "0–290 width, 0–55 height."
Second child → Widget: "I'll take 140x30."
```

```
Widget: "Positions: first at (5,5), second at (80,25). My total size:
300x60."
```

🧱 1: Exploring Basic Layout Widgets

1. **Create a new Flutter app** named `layout_lab`.
2. In `main.dart`, replace the body of your `Scaffold` with a simple `Container`.

```
Scaffold(
  appBar: AppBar(title: Text('Layout Lab')),
  body: Container(
    color: Colors.blueGrey[100],
    child: Center(
      child: Text('Hello Layout!'),
    ),
  ),
);
```

3. Experiment by wrapping the `Text` in:
    - a `Padding`
    - a `Row`
    - a `Column`
    - a `Stack`

Observe how constraints and alignment affect the layout.
👉 Try printing widget sizes using `LayoutBuilder` and `print()` statements.

🧩 2: Constraints in Action

Add a `LayoutBuilder` to inspect constraints:

```
LayoutBuilder(
  builder: (context, constraints) {
    print('Max width: ${constraints.maxWidth}');
    return Container(
      color: Colors.amber,
      width: 100,
      height: 100,
      child: Text('Inside constraints!'),
    );
  },
);
```

Try changing the parent widget (e.g., wrap in a `Center`, `Align`, or `Expanded`) and see how the constraints change.

⚖️ 3: Common Layout Patterns

Widget	Purpose	Example
`Row` / `Column`	Arrange children horizontally / vertically	Menu bars, lists
`Expanded` / `Flexible`	Let child fill available space proportionally	Dynamic grids
`Stack`	Overlap children	Cards, image overlays
`Align` / `Center`	Control position inside parent	Positioning logos
`SizedBox`	Set fixed or empty space	Spacing between widgets
`Container`	General-purpose styling + constraints	Borders, backgrounds

💡 **Exercise:**
Recreate this simple layout using combinations of `Row`, `Column`, and `Expanded`:

```
| Title              | Icon |
|--------------------|------|
| Description text...      |
| [Button 1] [Button 2]    |
```

🎨 4: Visual Exercise – Nested Constraints

Create this structure step-by-step:

```
Container(
  color: Colors.blueGrey,
  padding: EdgeInsets.all(20),
  child: Container(
    color: Colors.amber,
    width: double.infinity,
    height: 150,
    child: Center(
      child: Container(
        color: Colors.green,
        width: 100,
        height: 50,
        child: Text('Inner box'),
      ),
    ),
  ),
);
```

Then:

- Change inner container width to `double.infinity` → What happens?
- Remove the outer padding → What changes?

- Replace `Center` with `Align(alignment: Alignment.bottomRight)` → Observe constraints again.

🧠 Challenge Task

Build a **responsive card** that:

- Has a header (`Row`) with a title and icon.
- Contains body text with a fixed padding.
- Adapts its width to screen size using `LayoutBuilder`.

Optional: Try to add a `Stack` overlay (like a floating badge or notification dot).

🏁 Summary

- Constraints define the *space rules* of a widget.
- Size is decided by the widget itself (within its constraints).
- The parent positions each widget.
- Layout widgets are how you negotiate these three forces.