TIERS LIMITED SUMMER INTERNSHIP 2024

MOBILE APP DEVELOPMENT

Flutter Basics

CLASS # 3

Overview

In this class, we will cover the fundamental concepts of Flutter, focusing on understanding widgets, the widget tree, and basic Flutter layout concepts such as Container, Row, and Column. These are essential building blocks for creating user interfaces in Flutter.

Objectives

- Understand the fundamental concepts of widgets in Flutter.
- Learn the difference between stateless and stateful widgets.
- Understand the widget tree structure in a Flutter app.
- Gain knowledge about basic Flutter layout concepts such as **`Container`**, **`Row`**, and **`Column`**.
- Learn how to use **`MainAxisAlignment`** and **`CrossAxisAlignment`** for layout control.

Starting Point of a Flutter App

`main` Function

- **Purpose:** The **`main`** function is where execution of your Flutter/Dart application begins.
- **Role:** It calls the **`runApp`** function and passes an instance of your root widget (**`MyApp`**).

`runApp` Function

- **Purpose:** **`runApp`** is a Flutter function that takes the root widget of your application.
- **Role:** It initializes the app and starts the Flutter framework, which manages the app's lifecycle, rendering, and updates.

Root Widget (`MyApp`)

- **Purpose:** This is typically a **`StatelessWidget`** or **`StatefulWidget`** that represents the entire application.
- **Role:** It sets up fundamental aspects of the app, such as its title, theme, and initial screen (**`home`** property of **`MaterialApp`** or **`CupertinoApp`**).

Example Code

```
void main() {
  runApp(const MyApp());
}
```

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

In this example:

- `main` function calls `runApp` with an instance of `MyApp`.
- `MyApp` is a stateless widget defining the app's basic structure using `MaterialApp`.
- `MaterialApp` configures the app's visual properties like theme and initial screen.

Key Points:

- **Entry Point:** `main` function starts the Flutter app execution.
- `runApp`**:** Initializes Flutter framework with the root widget.
- **Root Widget (**`MyApp`**):** Defines the app's structure and initial settings.

Widgets

- Widgets are the building blocks of a Flutter app's user interface.
- Everything in Flutter is a widget, including layout models, controls, and animations.
- Widgets are nested inside one another to build complex UIs.

Types of Widgets

- **Stateless Widgets:** Immutable widgets that do not change their state once built.
- **Stateful Widgets:** Widgets that can change their state and redraw when the state changes.

Basic Flutter Layout Concepts

Scaffold:

- The `Scaffold` widget is a top-level container used in Flutter apps to provide a structure for the basic material design visual layout.
- The Scaffold widget is usually used as the root of the widget tree for a screen. It takes a single child, which is typically a **body** widget that fills the remaining space in the screen2. The body widget can be any widget that displays the main content of the screen.

Container:

The **`Container`** widget is a versatile widget that can hold a single child widget. It allows you to customize its appearance with properties such as padding, margin, alignment, and decoration.

```
Container(
  padding: EdgeInsets.all(10.0),
  margin: EdgeInsets.all(10.0),
  decoration: BoxDecoration(
    color: Colors.blue,
    borderRadius: BorderRadius.circular(10.0),
  ),
  child: CHILD 1,
);
```

Row and Column:

`Row` and **`Column`** are layout widgets that arrange their children horizontally and vertically, respectively.

Row:

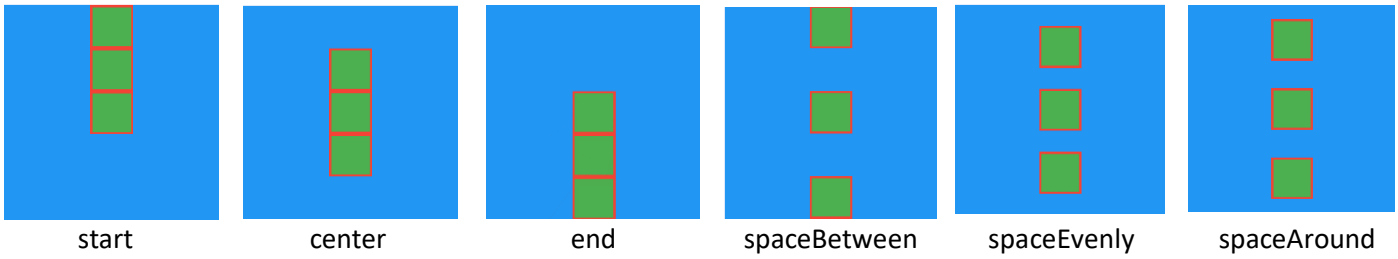
```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Child 1,
    Child 2,
    Child 3,
  ],
);
```

Column:

```
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Child 1,
    Child 2,
    Child 3,
  ],
);
```

MainAxisAlignment and CrossAxisAlignment:

- **`MainAxisAlignment`** aligns children along the main axis (horizontal for `Row`, vertical for `Column`).
- **`CrossAxisAlignment`** aligns children along the cross axis (vertical for `Row`, horizontal for `Column`).



The Widget Tree:

- The widget tree is a hierarchical representation of the widget structure in a Flutter app.
- Each widget can have children, leading to a tree structure.
- The root of the widget tree is typically the **`MaterialApp`** or **`CupertinoApp`** widget.

Exercises:

Exercise 1: Creating a Basic Layout

1. **Objective:** Practice using Container and Column widgets.
2. **Task:**
    - Create a Container with padding, margin, and decoration properties.
    - Inside the Container, create a Column with three nested Container widgets.

Exercise 2: Row and Column Alignment

1. **Objective:** Learn how to align widgets using Row and Column with MainAxisAlignment and CrossAxisAlignment.
2. **Task:**
    - Create a Row with three Container widgets.
    - Align the containers using MainAxisAlignment.spaceAround.
    - Create a Column with three Container widgets.
    - Align the containers using CrossAxisAlignment.center.

Exercise 3: Understanding the Widget Tree

1. **Objective:** Visualize and understand the hierarchical structure of widgets in a Flutter app.
2. **Task:**
    - Create a simple widget tree with MaterialApp, Scaffold, AppBar, and Center widgets.
    - Inside the Center widget, add a Column with two Container widgets.