Flutter Widgets: What is Container & How to Use it in Flutter?

Hello friends! In today's class, we are going to explore the Flutter `Container` widget, understand its usage, and see practical examples of how it can be implemented. Let's dive in!

1. Understanding the Container

What is a Container?

- A `Container` is an invisible box that can contain other widgets and arrange them in a specific layout.
- It is commonly used to group widgets together, apply styling, positioning, and size.

2. Properties of a Container

Key Properties

- `color`: Sets the background color of the container.
- `width` and `height`: Define the dimensions of the container.
- `child`: The single widget that this container holds.
- `decoration`: Allows you to apply various styles like background images, gradients, borders, etc.

3. Creating a Container

Basic Example

Here's a simple example of creating a `Container` in Flutter:

```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Container',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter Container Example'),
        ),
        body: Center(
          child: Container(
            width: 100,
            height: 100,
            color: Colors.blue,
            child: Center(
              child: Text(
                'Hello Flutter',
                style: TextStyle(color: Colors.white),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Explanation

- We define a `Container` with a width and height of 100.
- We set the `color` to blue.
- The `child` of the container is a centered `Text` widget displaying "Hello Flutter".

4. Advanced Container Usage

Using Decoration

You can enhance the appearance of a `Container` using the `decoration` property:

```
Container(
  width: 200,
  height: 200,
  decoration: BoxDecoration(
    color: Colors.red,
    borderRadius: BorderRadius.circular(10),
    boxShadow: [
      BoxShadow(
        color: Colors.black.withOpacity(0.5),
        spreadRadius: 5,
        blurRadius: 7,
        offset: Offset(0, 3), // changes position of shadow
      ),
    ],
  ),
  child: Center(
    child: Text(
      'Decorated Container',
      style: TextStyle(color: Colors.white),
    ),
  ),
)
```

Explanation

- `BoxDecoration` allows setting a background color, border radius, and box shadow.
- The `borderRadius` creates rounded corners.

- `boxShadow` adds a shadow effect to the container.

5. Practical Implementation

Step-by-Step Guide

1. **Create a New Flutter Project**: Use Android Studio or VS Code to create a new Flutter project.
2. **Set Up the Main Widget**: In `main.dart`, set up your main widget as shown in the basic example.
3. **Customize the Container**: Modify the `Container` to use various properties like `width`, `height`, `color`, `decoration`, etc.

Example Project

```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Container',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter Container Example'),
        ),
        body: Center(
          child: Container(
            width: 200,
            height: 200,
            decoration: BoxDecoration(
              color: Colors.green,
              borderRadius: BorderRadius.circular(15),
              border: Border.all(
                color: Colors.black,
                width: 4,
              ),
            ),
            child: Center(
              child: Text(
                'Hello Flutter',
```

```
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 20,
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Explanation

- We create a `Container` with specific dimensions, background color, rounded corners, and a border.
- The `child` of the container is a `Text` widget centered within the container.

6. Conclusion

- The `Container` widget in Flutter is a versatile tool for layout and styling.
- It can hold a single child, which can be a complex layout with multiple nested widgets.
- Using properties like `color`, `width`, `height`, and `decoration`, you can create visually appealing designs.

By understanding and utilizing the `Container` widget, you can enhance your Flutter applications with more organized and stylish UIs. Happy coding!