Understanding FutureBuilder Widget in Flutter

Asynchronous Programming in Flutter

Introduction

- Definition: The FutureBuilder widget allows you to build widgets based on the latest snapshot of interaction with a Future.
- Purpose: To simplify the process of handling asynchronous data in Flutter apps.

Why Use FutureBuilder?

- Asynchronous Programming: Handling data that may not be available immediately (like API calls).
- State Management: Automatically updates the UI based on the state of the Future.
- Error Handling: Provides a way to manage loading, success, and error states in the UI.

```
FutureBuilder<DataType>(
  future: yourFutureFunction(),
  builder: (BuildContext context,
AsyncSnapshot<DataType> snapshot) {
    if (snapshot.connectionState ==
ConnectionState.waiting) {
      return CircularProgressIndicator(); // Loading state
    } else if (snapshot.hasError) {
      return Text('Error: ${snapshot.error}'); // Error state
    } else {
      return YourWidget(data: snapshot.data); // Success
state
    }
  },
);
```

FutureBuilder Properties

future: The Future you want to retrieve data from.

builder: A function that builds the UI based on the AsyncSnapshot.

connectionState: An enumeration that indicates the state of the connection to the future.

Connection States

- ConnectionState.none: No connection to the
- future.ConnectionState.waiting: The future is still
- running.ConnectionState.active: The future is actively
- running.ConnectionState.done: The future has completed.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: FutureBuilderExample(),
    );
  }
}
class FutureBuilderExample extends StatelessWidget {
  Future<String> fetchData() async {
    // Simulate a 2-second delay
    await Future.delayed(Duration(seconds: 2));
    // Uncomment the next line to simulate an error
    // throw Exception('Failed to load data');
    return 'Data fetched successfully!';
  }
```

```
 @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('FutureBuilder Example'),
      ),
      body: Center(
        child: FutureBuilder<String>(
          future: fetchData(),
          builder: (context, snapshot) {
            // Show a loading spinner while waiting for the future
to complete
            if (snapshot.connectionState ==
ConnectionState.waiting) {
              return CircularProgressIndicator();
            }
else if (snapshot.hasError) {
              return Text('Error: ${snapshot.error}');
            }
else if (snapshot.hasData) {
              return Text(snapshot.data!);
            }else {
              return Text('No data available');
            }
          },
        ),
      ),
```

Error Handling in FutureBuilder

```
if (snapshot.hasError) {
  return Text('Error: ${snapshot.error}');
}
```