Flutter Networking: Fetching API Data

Subtitle: Learn how to connect Flutter apps to APIs and display dynamic data.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Most apps need to **fetch data from the internet**. Flutter makes it easy using the `http` package to request APIs and display dynamic content. This guide covers fetching, parsing, and displaying data.

Step 1: Add HTTP Dependency

```
dependencies:
  http: ^1.1.0
```

Run:

```
flutter pub get
```

Import in Dart:

```
import 'package:http/http.dart' as http;
import 'dart:convert';
```

Step 2: Fetch Data from API

```
Future<void> fetchData() async {
  final response = await http.get(Uri.parse('https://jsonplaceholder.typicode.co
  if (response.statusCode == 200) {
    final data = jsonDecode(response.body);
    print(data);
  } else {
    print('Error: ${response.statusCode}');
  }
}
```

Exercise: Call `fetchData()` in `initState` and print the results.

Step 3: Display Data in ListView

```
FutureBuilder(
  future: fetchData(),
  builder: (context, snapshot) {
    if (!snapshot.hasData) return CircularProgressIndicator();
    final posts = snapshot.data as List;
    return ListView.builder(
      itemCount: posts.length,
      itemBuilder: (context, index) {
        return ListTile(
          title: Text(posts[index]['title']),
          subtitle: Text(posts[index]['body']),
        );
      },
    );
  },
)
```

Exercise: Display posts from JSONPlaceholder API in a scrolling list.

Step 4: Error Handling

```
try {
  final response = await http.get(Uri.parse(url));
  if (response.statusCode != 200) throw Exception('Failed to load data');
} catch (e) {
  print('Error: $e');
}
```

Exercise: Handle network errors and show a friendly message to users.

Step 5: Mini Project

Build a **Flutter API App**:

- Fetch posts from JSONPlaceholder
- Display posts in `ListView` with title and body
- Show loading spinner while fetching

- Handle errors gracefully

Key Takeaways

- `http` package fetches and parses API data
- `FutureBuilder` helps display data asynchronously
- Proper error handling improves **UX and reliability**
- Networking is essential for **dynamic, real-world apps**

Skipping networking makes your app **static and unable to access live content**, limiting its usefulness.

Visit **haas.dev** for more Flutter guides, tutorials, and complete project resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app
