Flutter State Management: Provider Basics for Beginners

Subtitle: Learn how to manage app state efficiently using Provider in Flutter.

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

Introduction

Managing state in Flutter apps is crucial for **dynamic UI updates and data flow**. Provider is a beginner-friendly, efficient way to handle state. This guide teaches the basics and practical usage.

Step 1: Add Provider Dependency

```
dependencies:
  provider: ^6.0.5
```

Run:

```
flutter pub get
```

Import in Dart:

```
import 'package:provider/provider.dart';
```

Step 2: Create a ChangeNotifier Class

```
class Counter with ChangeNotifier {
  int _count = 0;

  int get count => _count;

  void increment() {
    _count++;
    notifyListeners();
  }
}
```

- `notifyListeners()` updates UI when state changes

Step 3: Wrap App with ChangeNotifierProvider

```
void main() {
  runApp(
    ChangeNotifierProvider(
      create: (context) => Counter(),
      child: MyApp(),
    ),
  );
}
```

Exercise: Wrap your main app with Provider to manage counter state.

Step 4: Access State in Widgets

```
final counter = Provider.of<Counter>(context);
Text('Count: ${counter.count}');
ElevatedButton(
  onPressed: () {
    counter.increment();
  },
  child: Text('Increment'),
)
```

Exercise: Create a button that increases counter and updates UI instantly.

Step 5: Using Consumer Widget

```
Consumer<Counter>(
  builder: (context, counter, child) {
    return Text('Count: ${counter.count}');
  },
)
```

- `Consumer` rebuilds only the widgets that need state changes
- Improves performance for large apps

Step 6: Mini Project

Build a **Counter App with Provider**:

- Increment and decrement buttons
- Display updated counter in multiple widgets
- Use `Consumer` for selective rebuilding

Key Takeaways

- Provider is beginner-friendly and efficient for state management
- `ChangeNotifier` + `notifyListeners()` keeps UI in sync
- `Consumer` optimizes performance
- Proper state management is **essential for scalable Flutter apps**

Skipping state management makes your app **fragile, hard to maintain, and slow** as it grows.

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

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