Flutter Development Guide for Beginners

Build Beautiful Cross-Platform Mobile Apps with Dart

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

Introduction

Flutter allows developers to create **native Android and iOS apps using a single codebase in Dart**.
This guide covers **setup, widgets, state management, navigation, and mini-projects** for beginners.

---

1. Environment Setup

- Install **Flutter SDK**
- Install **Android Studio** (for Android emulator)
- Install **Xcode** (for iOS simulator, macOS only)
- Install **VS Code** (optional)
- Verify installation: `flutter doctor`

---

2. Project Structure

- **lib/** – Main source code
- **main.dart** – App entry point
- **widgets/** – Reusable UI components
- **screens/** – App pages
- **assets/** – Images, fonts, icons

---

3. Core Concepts

Widgets

- Everything in Flutter is a widget
- Types:

  - **StatelessWidget** – UI doesn't change dynamically

- **StatelessWidget** – UI doesn't change dynamically
- **StatefulWidget** – UI changes based on state

Dart Basics

- Variables, functions, loops, classes
- Object-oriented programming principles

---

4. Layouts and Styling

- **Container, Row, Column, Stack** – Core layout widgets
- **Padding, Margin, Alignment** – Styling and positioning
- **ThemeData** – Global app styling

```
Container(
  padding: EdgeInsets.all(10),
  child: Text('Hello Flutter'),
)
```

---

5. Navigation

- **Navigator & Routes** – Navigate between screens
- Named routes vs. direct routes
- Example:

```
Navigator.push(context, MaterialPageRoute(builder: (context) => SecondPage()));
```

---

6. State Management

- **setState()** – Simple local state
- **Provider** – Recommended for medium apps
- **Bloc / Riverpod** – For complex, scalable apps

---

7. API Integration

- Use **http** package

- Async calls with `Future` and `async/await`

```
final response = await http.get(Uri.parse('https://api.example.com'));
```

- Parse JSON using `jsonDecode`

---

8. Testing & Debugging

- Flutter DevTools for UI inspection
- Unit Testing: `flutter_test`
- Widget Testing: Ensure UI renders correctly

---

9. Deployment

Android

- Generate APK or AAB
- Upload to Google Play Store

iOS

- Build IPA
- Upload to Apple App Store

---

10. Mini Project Ideas

1. **To-Do List App** – CRUD operations, local storage
2. **Weather App** – API integration, UI cards
3. **Expense Tracker** – Charts, state management
4. **News Reader App** – API integration, infinite scroll

Best Practices

- Use modular widgets for reusability
- Keep UI responsive for multiple screen sizes
- Apply proper state management

- Optimize performance for smooth animations

Key Takeaways

- Flutter allows **beautiful cross-platform apps** with a single codebase
- Widgets, state management, and navigation are fundamental
- API integration, testing, and deployment skills are essential
- Mini-projects accelerate learning and portfolio building

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