ScopeTreasure Courses

FLUTTER COURSES · LESSON F.2 · CODE SCOPE SERIES

Flutter Setup — Pehla Project Chalao

Install the Flutter SDK, set up VS Code, pass flutter doctor, create your first project and feel hot reload.

Introduction

Today Flutter moves from idea to your own computer. You will install the three things every Flutter developer has — the **Flutter SDK** (the toolkit itself), **VS Code** with the Flutter and Dart extensions (where you write), and a **device to run on** (an Android emulator or your real phone) — then create your first project and run it. The setup takes patience once; after today, starting work means opening a folder. *(Roman Urdu: Setup aik dafa ka kaam hai — SDK, VS Code, aur phone/emulator, bas.)*

Key points

- Download the **Flutter SDK** from flutter.dev and unzip it to a simple folder like `C:\src\flutter`. Then add its `bin` folder to your system **PATH** (environment variables) — PATH is how your terminal finds the `flutter` command from any folder. Without this step, every command answers "not recognized".
- In **VS Code**, install the **Flutter** extension by Dart Code (it brings the **Dart** extension with it). These two add syntax colours, error checking, widget helpers and the run button.
  (Roman Urdu: Extensions ke baghair VS Code sirf aik notepad hai — Flutter extension hi use dimagh deti hai.)
- **`flutter doctor`** is your health report. Run it in a terminal; it checks the SDK, the Android toolchain, Chrome and VS Code, and prints a tick `[✓]` or a cross for each — plus *instructions to fix* whatever is missing. Do not fear the crosses; doctor tells you the medicine too.
- **`flutter create my_first_app`** builds a complete starter project (named in small letters with underscores). Inside it, your file is `lib/main.dart` — almost all your writing happens in the `lib` folder.
- **`flutter run`** (from inside the project folder) compiles the app onto the connected emulator or phone. The first run takes minutes; later runs are fast.
- **Hot reload** is Flutter's magic while the app runs: change code, save (Ctrl + S), press `r` in the terminal (or the lightning button in VS Code), and the phone updates in about a second *without losing its state*. **Hot restart** (capital `R`) starts the app fresh when reload is not enough.

Code examples

Example 1 — Meet the doctor

```
flutter doctor
```

What it does: One command checks your whole setup and prints a "Doctor summary". This is the first command to run whenever Flutter "mysteriously" stops working — nine times out of ten, doctor names the

problem and its fix in plain words.

What you will see: lines like these (versions and details vary by computer):

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.x.x)
[✓] Android toolchain - develop for Android devices
[✓] Chrome - develop for the web
[✓] Visual Studio Code
• No issues found!
```

The last line — "No issues found!" — is your green signal. Crosses are not failure; they are a to-do list, and each one arrives with its own instructions.

Example 2 — Create and run your first project

```
flutter create my_first_app
cd my_first_app
flutter run
```

What it does: `flutter create` generates the starter app (the famous counter demo). `cd` steps inside the project folder — `flutter run` only works from inside a project. The run command builds the app and installs it on the emulator or phone you started first.

What you will see: after a first build that takes a while, the counter app appears on the device: a title bar, a big "0", and a round **+** button. Tap the button and the number grows — you did not write this app, but by the end of Lesson F.3 you will be able to explain every line of it.

Example 3 — The heart of the file flutter create gives you

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Counter App')),
        body: const Center(
          child: Text('0', style: TextStyle(fontSize: 40)),
        ),
      ),
    );
  }
}
```

What it does: This is a *shortened* view of the real `lib/main.dart` — the counter's full file is longer, but its skeleton is exactly this: `main()` runs `MyApp`, a **class** (D.8) that *extends* `StatelessWidget` (F.3 explains), whose `build` method returns the widget tree from Lesson F.1 — MaterialApp, Scaffold, AppBar, Center, Text.

What you will see: the same counter screen frame: the "Counter App" title bar and the number in the centre. Now do the hot-reload ritual once: change `'Counter App'` to `'My First App'`, save, press `r` — the title bar changes while the app keeps running. That one-second loop is how Flutter developers spend their days.

Quick recap

- Three things: Flutter SDK (+ PATH), VS Code + Flutter/Dart extensions, emulator or real phone.
- `flutter doctor` = health report with fixes. Run it first when anything breaks.
- `flutter create name` → `cd name` → `flutter run`. Your code lives in `lib/main.dart`.
- Hot reload (`r`) updates the running app in ~1 second; hot restart (`R`) starts it fresh.

Common mistakes

1. **Skipping the PATH step** — `flutter` answers "not recognized" in every folder. The command only exists for your terminal after `flutter\bin` is added to PATH and the terminal is restarted. Ninety percent of "Flutter is broken" is this one step.

2. **Running `flutter run` from the wrong folder** — it must run *inside* the project (the folder that contains `pubspec.yaml`). One `cd` too few and Flutter says it cannot find a project.
3. **Expecting hot reload to apply every change** — reload refreshes the UI code, but changes to `main()`, new fields, or native settings need a hot restart (`R`) or a full re-run. When a change "does nothing", restart before you debug.

Practice

1. Install the SDK and extensions, run `flutter doctor`, and fix every cross by following doctor's own instructions until it reports "No issues found!" (or only crosses you understand and have a plan for).
2. Create a project named `mera_app`, run it on an emulator or your phone, then change the title text in `lib/main.dart` to your name and hot reload. Screenshot the before and after — that pair is your setup trophy.

Practice quiz

1. What does flutter doctor do?

- It fixes your code automatically
- It is a health report: it checks the SDK, Android toolchain, Chrome and VS Code, with a tick or a cross and fix instructions for each
- It deletes broken projects
- It installs VS Code

Answer: It is a health report: it checks the SDK, Android toolchain, Chrome and VS Code, with a tick or a cross and fix instructions for each

2. After flutter create my_first_app, where does almost all your writing happen?

- In the android folder
- In lib/main.dart
- In pubspec.yaml only
- In the README file

Answer: In lib/main.dart

3. What is the difference between hot reload (r) and hot restart (R)?

- There is no difference
- Hot reload updates the running app in about a second without losing state; hot restart starts the app fresh
- Hot reload restarts fresh, hot restart keeps state
- Hot reload works only on iOS

Answer: Hot reload updates the running app in about a second without losing state; hot restart starts the app fresh

Watch the video lesson: https://youtu.be/cirbynqHjEA

Read online: https://scopetreasure.com/notes/flutter-setup-first-project · More free tools and notes at https://scopetreasure.com