

# Self teaching myself Flutter, part 1: Introduction and installation

---

This series of articles is meant as an introductory course on learning the Flutter framework. Rather than teaching the language itself or the framework in depth, I will be retracing all the steps I took while making my first mobile application, with some of the roadblocks and peculiarities I encountered along the way. That way, someone who is brand new to Flutter app development will be able to follow along, and avoid tripping on the same bumps I did.

## A short self-introduction

I am currently a 4th-year student at Supinfo International University, my experience lies mostly in backend development using NestJS, and a general knowledge of NodeJS. I've mostly been studying things relating to the web and app development in high-level languages.

This series will assume the reader has a general knowledge of web development, such as what an API is, how to call one, HTTP requests, etc. I will also occasionally make parallels between Flutter mobile development and more popular frontend frameworks like React to make some things easier to conceptualize for readers that have some experience using NodeJS.

## Why I picked flutter

The school I'm in encourages students to engage in a free-form project each year, one such project that I had in mind was an upgrade of a very simple mobile app I made back in 2023 for my mother's birthday.

The concept is simple, a Google Sheets document is maintained, containing a list of songs, and each day, a mobile application (made in Kotlin) downloads the next sheet and displays the song of the day. It was made in about four days.

Now, my objective was to make a new version of this concept, but with more features such as:

- Authentication ;
- Managing multiple calendars and daily messages directly within the app ;
- Collaborating with other people on the same calendar ;
- Othe functional and quality-of-life improvements.

With these new features in mind, I knew Kotlin was not going to cut it, the developer experience I had with it was simply not good enough to consider it for a larger-scale project.

Having had some experience with React during my apprenticeship, I considered React-Native as a possible candidates, but I also vaguely knew about something called "Flutter" at the time.

The reason I ended up choosing flutter is simple, it was mere curiosity. I would challenge myself with a completely new framework, and hopefully it will be a good experience.

## A primer on Dart

The 1.0 version of dart came out in 2013, google intended for it to replace JavaScript entirely. It has strong typing by default, making it easy to pick up for those who are used to Typescript.

Dart has its own [package manager](#), not unlike npm.

While the repository is not as thriving, it benefits from having some [officially maintained packages](#) with higher quality standards. Being more recent, and having a mission statement of "replacing Javascript", means that the team behind Dart is particularly cautious of not reproducing some of the same mistakes node could have over the years.

Nowadays, Dart is very strongly associated to Flutter, you generally don't hear much about it outside of this specific use case.

## Installing Flutter

The core installation of Flutter is a [fairly easy and well detailed process](#), all in all, it's just about installing dart, and then the Flutter SDK can be installed via the VSCode extension.

The harder part is getting the whole Android development environment setup.

Once you run `flutter doctor` on a clean installation, you'll likely be warned that the Android SDK cannot be found. To use it. The only way to get it is by installing the Android Studio IDE, even if you are not planning on using it.

On Linux, it may be possible that flutter is looking for a different location, by setting the `ANDROID_SDK_ROOT` in your environment (like with the `export` command in `.bashrc`), you should get `flutter doctor` to complete without any warning.

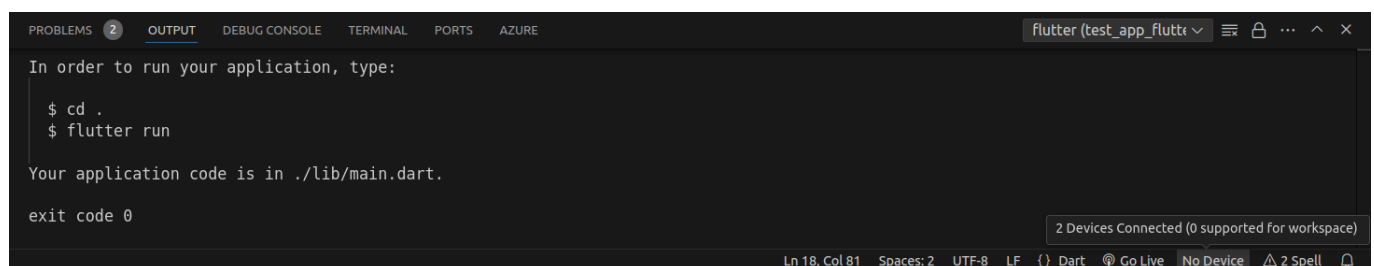
Note: if you are planning to develop for IOS, then it is not Android Studio, but XCode that needs to be fully set up and configured.

## Creating a flutter project

Once your development environment is fully configured, you can bring up the command palette `ctrl+shift+P` and use `Flutter: New Project` to create your first flutter project.

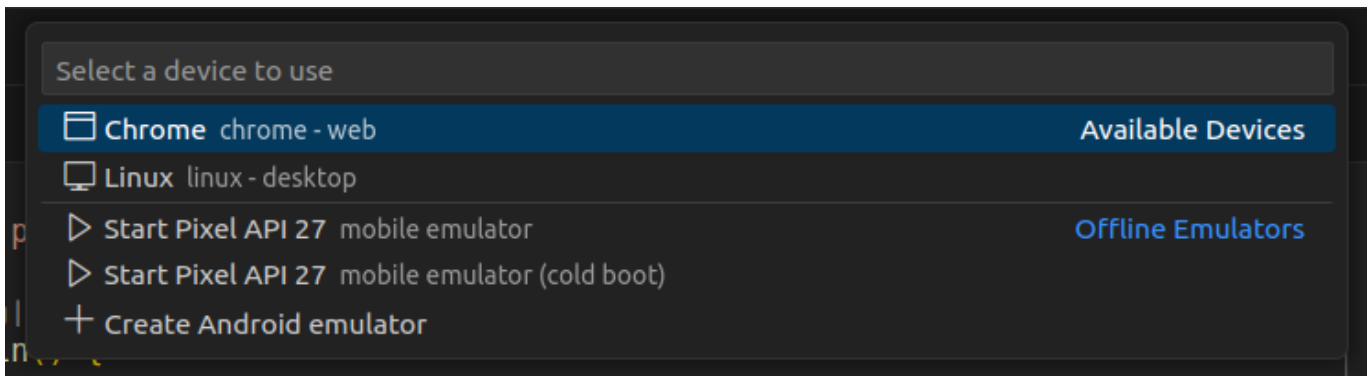
## Launching the emulator

Now that a mobile development environment is set up on your machine. VSCode will recognize flutter projects and give you a "device selection" option.



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE flutter (test_app_flutter)
In order to run your application, type:
$ cd .
$ flutter run
Your application code is in ./lib/main.dart.
exit code 0
2 Devices Connected (0 supported for workspace)
Ln 18, Col 81 Spaces: 2 UTF-8 LF {} Dart @ Go Live No Device 2 Spell
```

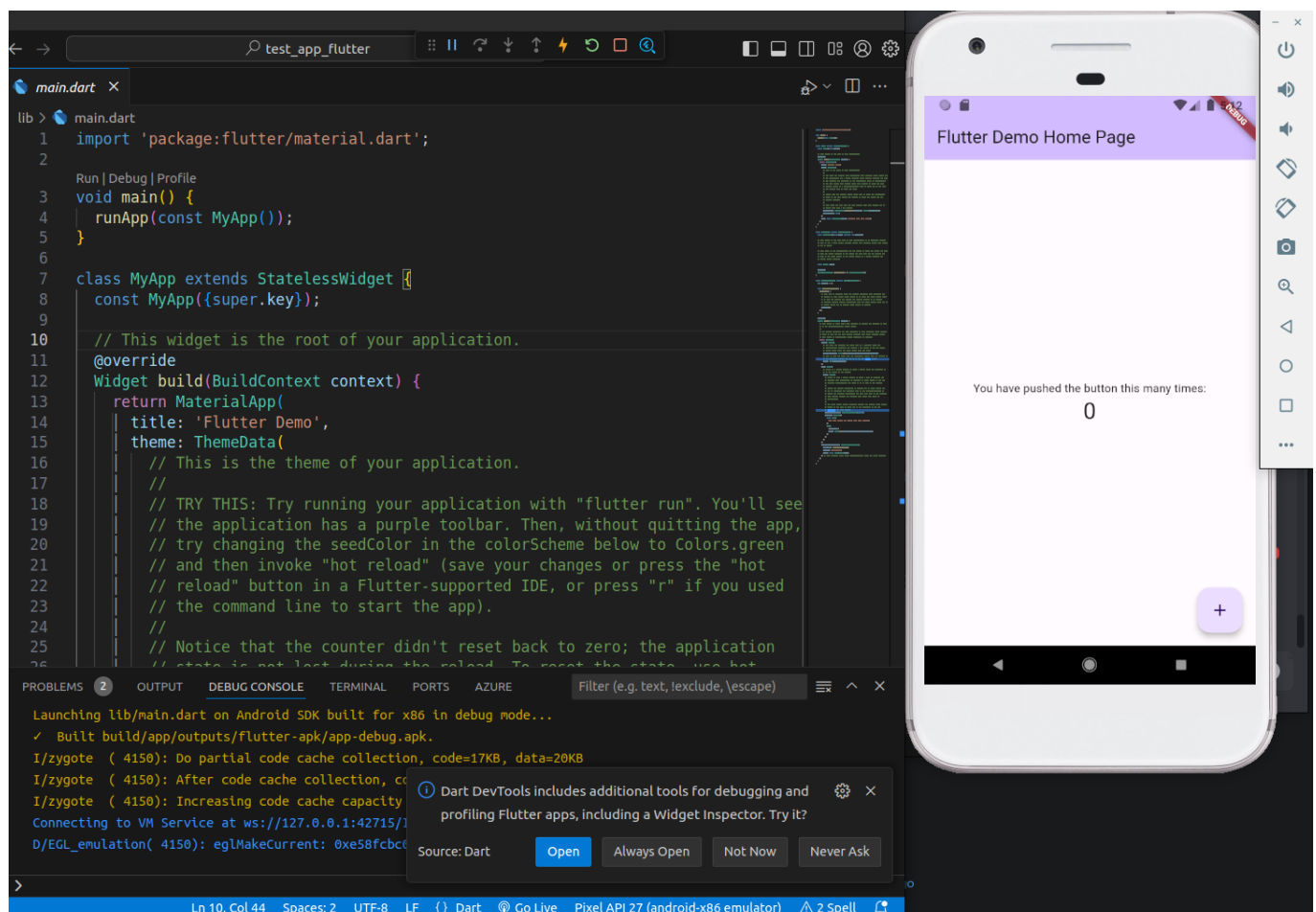
When clicking on this, you'll be prompted to select an emulator or create a new one. Emulators are managed by the android SDK and can also be managed in Android Studio.



Ensure the emulator that is running is the want you want flutter to build your app to by running `Flutter: Select Device` in VSCode's command palette.

Once you're set up, running `flutter run` at the root of your project or pressing `f5` in VSCode will launch your default app.

Once your application is built and ready, it will be run automatically by your emulator.



That's it! You are now ready to start making your Flutter app.

In the next part of this series, I will mention my own learning path through the basics, present some fundamental components of flutter as well as talk about the importance of tooling.