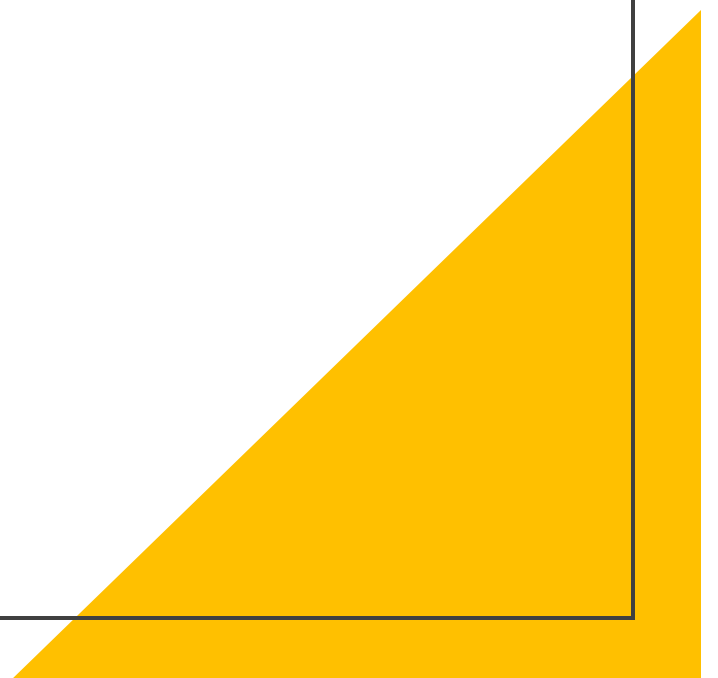


Getting Started with Flutter


By Minasie L.



Outlines

- How to use Git to manage the Flutter SDK
- Setting up the command line and saving path variables
- Using Flutter Doctor to diagnose your environment
- Configuring the Android SDK setup
- Which IDE/editor should you choose?
- How to create a Flutter app
- How Flutter projects are structured
- How to run a Flutter app
- How to use Hot reload to refresh your app without recompiling

Technical requirements

- 8 GB of random-access memory (RAM) (16 gigabytes (GB) preferred)
 - 50 GB of available hard drive space
 - A solid-state drive (SSD) hard drive is recommended
 - At least a 2 gigahertz (GHz) + processor
 - If you want to build for iOS, you will also need a Mac instead of a PC.
 - These are not strict system requirements, but anything less than this may lead to you spending more time waiting rather than working.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

How to use Git to manage the Flutter SDK

- Installing Git
 - For Windows, you can download and install Git here: <https://git-scm.com/download/win>
 - You might also want to get a Git client to make working with repositories a bit easier. Tools such as Sourcetree (<https://www.sourcetreeapp.com>) or GitHub Desktop (<https://desktop.github.com>) can greatly simplify working with Git.
- To confirm that Git is installed on Linux and macOS, if you open your Terminal and type `which git` , you should see a `/usr/bin/git` path returned.

How to do it

- First, choose a directory where Flutter is going to be installed. The location does not explicitly matter, but it will be simpler to install the SDK closer to the root of your hard drive.
- On Linux, type in the following command:
 - `cd $HOME`
- We can now install Flutter with this command:
 - `git clone https://github.com/flutter/flutter.git`



Setting up the command line and saving path variables

- In Linux
 - `export PATH="$PATH:$HOME/flutter/bin"`
-

Using Flutter Doctor to diagnose your environment



Flutter comes with a tool called Flutter Doctor that will be your new best friend when setting up the SDK.



Flutter Doctor will give you a list of everything that needs to be done to make sure that Flutter can run correctly.



You are going to use Flutter Doctor as a guide during the installation process.



This tool is also invaluable to check whether your system is up to date.



In your terminal window, type the following command:

```
flutter doctor
```

Configuring the Android SDK setup

-
- You can download Android Studio at <https://developer.android.com/studio>.
 - After Android Studio is installed, you'll need to download at least one Android SDK. From the Android Studio menu, select Preferences and then type android into the search field.
 - You will also need to download the latest build tools, emulator, SDK platform tools, SDK tools, the Hardware Accelerated Execution Manager (HAXM) installer, and the support library.
 - Select the SDK Tools tab and make sure the required components are checked. When you click the Apply or OK buttons, the tools will begin downloading.

Creating an Android emulator

- Select the Android Virtual Device Manager (AVD Manager) from the toolbar in Android Studio.
- The first time you open the AVD Manager, you'll get a splash screen. Select the Create Virtual Device... button in the middle to start building your virtual device.
- The next text screen allows you to configure which Android hardware you want to emulate. I recommend using a Pixel device.
- In the next screen, you will have to pull down an Android runtime. For the most part, the most recent image will be sufficient. Each one of these images is several gigabytes (GB) in size, so only download what you need

Creating an Android emulator

- Click Next to create your emulator. You can launch the emulator if you want, but this is not necessary.
- Once again, run flutter doctor to check your environment.
- One final thing that you may have to do is accept all the Android license agreements. You can do this quickly from the terminal line with this command:
 - `flutter doctor --android-licenses`

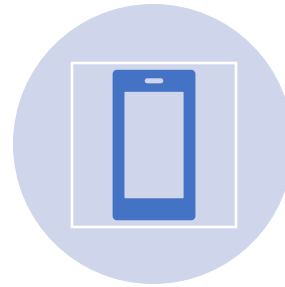
Which IDE/editor should you choose?

-
- Flutter provides official plugins for three popular IDEs:
 - Android Studio
 - **Visual Studio Code (VS Code)**
 - IntelliJ IDEA

How to create a Flutter app



There are two main ways to create a Flutter app: either via the command line or in your preferred IDE.



We're going to start by using the command line to get a clear understanding of what is going on when you create a new app.



Before you begin, it's helpful to have an organized place on your computer to save your projects. This could be anywhere you like, as long as it's consistent



So, before creating your apps, make sure you have created a directory where your projects will be saved.

How to do it



Flutter provides a tool called flutter create that will be used to generate projects.



Let's type this command to generate our first project:

```
flutter create hello_flutter
```

This command assumes you have an internet connection since it will automatically reach out to the public website to download the project's dependencies.



If you don't currently have an internet connection, type the following instead:

```
flutter create --offline hello_flutter
```



How to do it

- You'll need to either connect a device to your computer or spin up an emulator.
 - flutter emulators
 - You should see a list of available emulators
 - flutter emulators --launch [your device name, like: Nexus_5X_API_28]
 - cd hello_flutter
 - flutter run
 - For physical devices, in order to see all the connected devices, run the following command:
 - flutter devices
-

How to do it

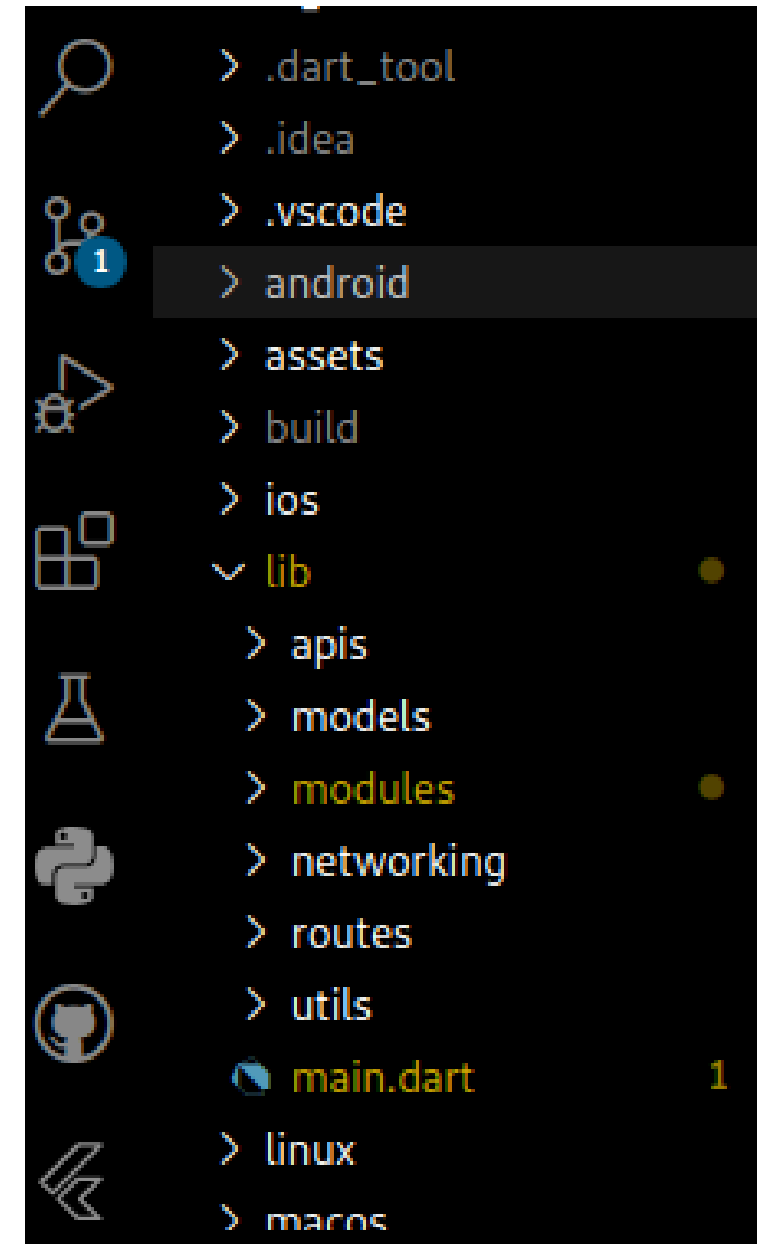
- To run your app on one of the available devices, type the following command:
 - `flutter run -d [your_device_name]`
- After your app has finished building, you should see a demo flutter project running in your device.

How to choose a platform language for your app

- Both iOS and Android are currently in the middle of a revolution of sorts.
- When both platforms started over 10 years ago, they used the Objective-C programming language for iOS, and Java for Android.
- These are great languages, but sometimes they can be a little long and complex to work with.
- To solve this, Apple has introduced Swift for iOS, and Google has adopted Kotlin for Android.

Where do you place your code?

- The files that Flutter generates when you build a project should look something like this:





Main folders

- The main folders in your projects are listed here:
 - Android
 - Build
 - ios
 - lib
 - Test
-



Thank You!
Do you have any question?

