PRACTICAL 9

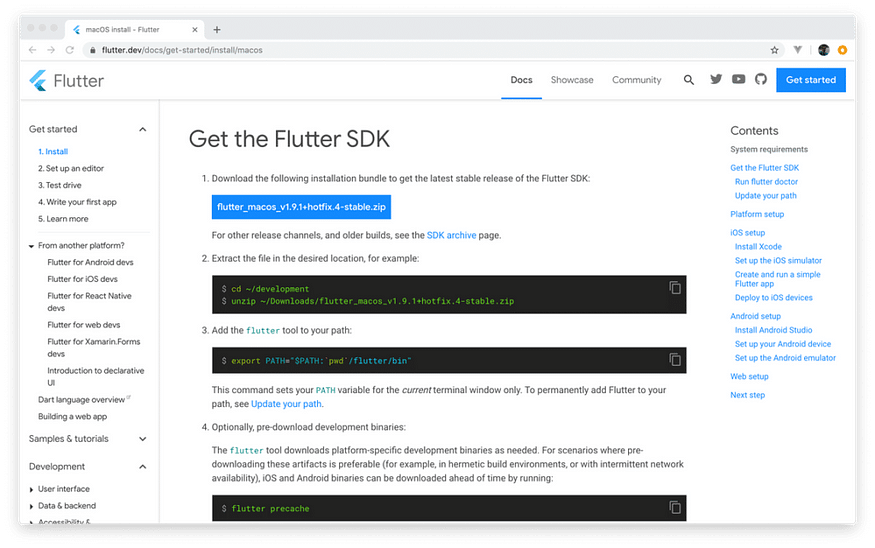
AIM: Configure Flutter Development environment.

1. **Download And Install The Flutter SDK**

First step is to setup the Flutter development environment. The SDK is available for the following platforms:

1. Windows
2. macOS
3. Linux

* The installation procedure depends on the platform you’re working on.
* To install the Flutter SDK on you first need to download the SDK from the link <https://flutter.dev/docs/get-started/install/>.



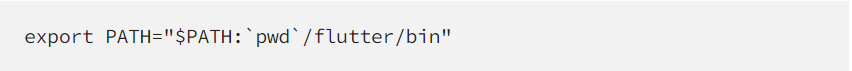
* Click on the blue button to download the latest stable Flutter SDK release as a ZIP archive. The ZIP file which is being downloaded includes the current version number of the Flutter SDK in its name, e.g. *flutter\_macos\_v1.9.1+hotfix.4-stable.zip*.
* The next step is to extract the archive to a location on your computer in which you’d like to store the flutter SDK files. E.g. you can create a new folder *development* in your user’s directory and extract the content of the archive in this location:

$ cd ~  
$ mkdir development  
$ cd development  
$ unzip ~/Downloads/filename

Inside the folder *development* you should now find a subfolder *flutter* which contains the files of the Flutter SDK.

1. **Adding Flutter SDK To Your Path**

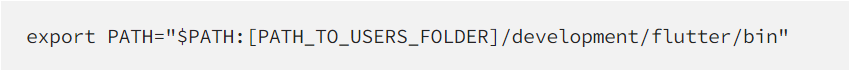
Next we need to add the */flutter/bin* folder to the *PATH* variable to be able to execute the *flutter* command from any location. This can be done by using the following inside of the *development* directory

****

* Using the export command sets the *PATH* for the current terminal session. This means that after opening up another terminal session, the export command has to be executed again to make the *flutter* command available.

1. **Setting The Path Permanently**

* In most cases you’d like to add the Flutter SDK path permanently to your PATH variable, so that you’re able to use the *flutter* command in any terminal window. Adding the path permanently can be done by performing the following actions.
* Open file *.bash\_profile* (located in your user’s folder) and add the following line at the bottom:

****

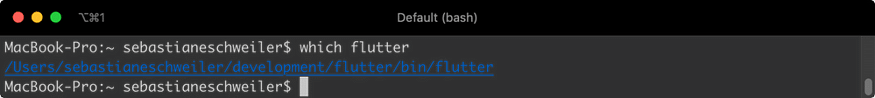
After restarting the terminal you know should be able to execute the *flutter* command from any location:



You can also check for the location from which *flutter* is used by using the *which* command in the following way:

$ which flutter

You should then be able to see an output which is similar to the following screenshot:



1. **Flutter Doctor**

* Now that we’ve added to the PATH we’re ready to use the flutter command for various purposes. By using the *doctor* option you can check if we need to install further dependencies

$ flutter doctor

**Setup Emulators**

When developing Flutter apps for iOS or Android you need to setup corresponding emulators on your system first. By using these virtual mobile devices you can run those apps easily when developing.

**Android**

* In order to develop Flutter apps for Android you need to make sure to perform a full installation of Android studio (<https://developer.android.com/studio>) on your system. This installs the latest Android SDK, Android SDK Platform-Tools, and Android SDK Build-Tools, which are required by Flutter when developing for Android.
* In order to setup an Android emulator you need to use the integrated AVD Manager to create an Android Virtual Device which then can be started as an emulator on your system.

1. **Setting Up Visual Studio Code**

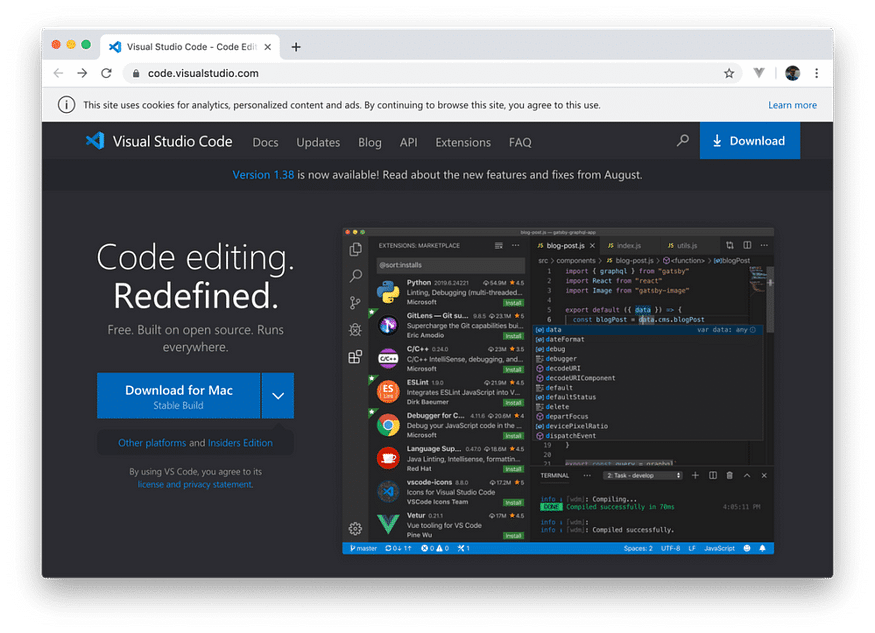
* To develop Flutter application with Dart in general you can use any code editor you like.

However there are two recommendations:

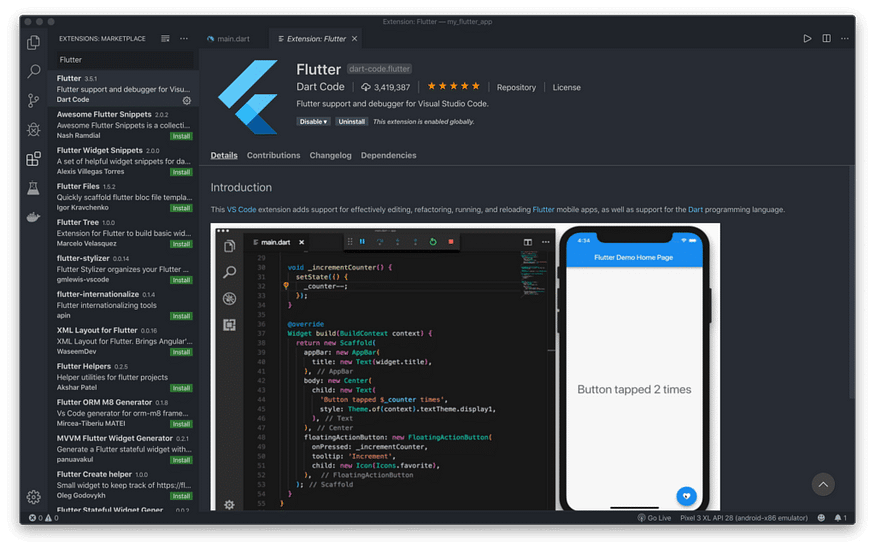
* Visual Studio Code
* Android Studio / IntelliJ

Both options supports writing Flutter applications with Dart in many ways. In the following we’re going to setup Visual Studio Code.

First you need to make sure to download the latest version of Visual Studio Code for you platform from <https://code.visualstudio.com/>.



Having downloaded and installed Visual Studio Code you then need to make sure to add the Flutter extension to the editor. Switch to the extension tab and search for “flutter”:



The first entry and in the search result list is the extension we’re looking for. By installing this extension we’re adding support for editing, refactoring, running, and reloading Flutter application, as well as support for the Dart programming language.

**Create And Run Your First Flutter**

* Now that everything is setup we’re ready to create the first Flutter project from scratch. The first option is to create the new project on the command line by using the *flutter* command once again:

$ flutter create my\_flutter\_app

* By using the option create we’re telling flutter to create a new project. The name of the new project folder is specified by using the second parameter. After the project is created you can change into the newly created project folder:

$ cd my\_flutter\_app

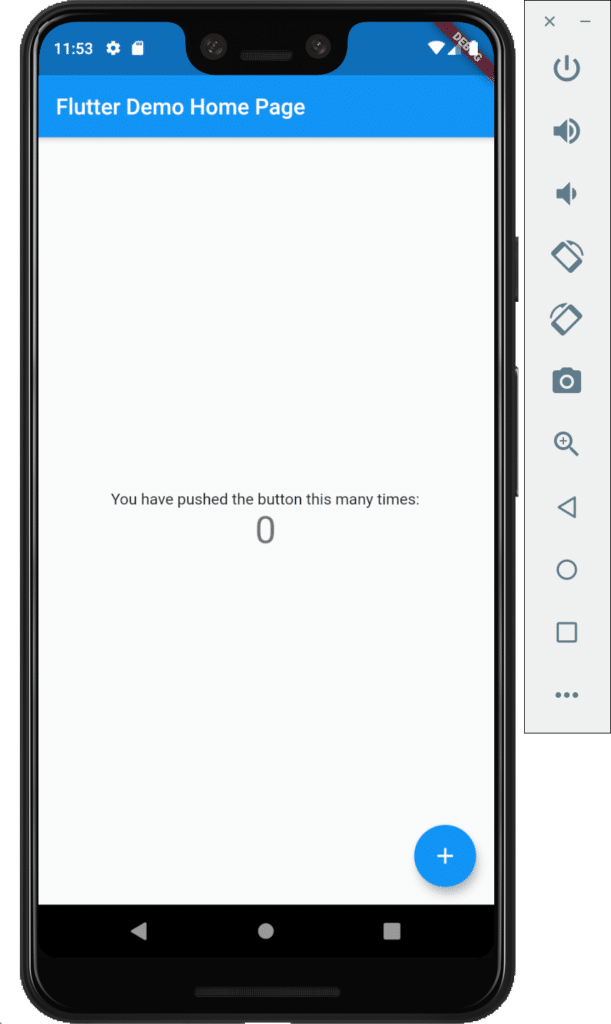
* Within this directory you can use the flutter command with option run to start the default flutter application in the simulator (the simulator must be started before, so that it is already running):

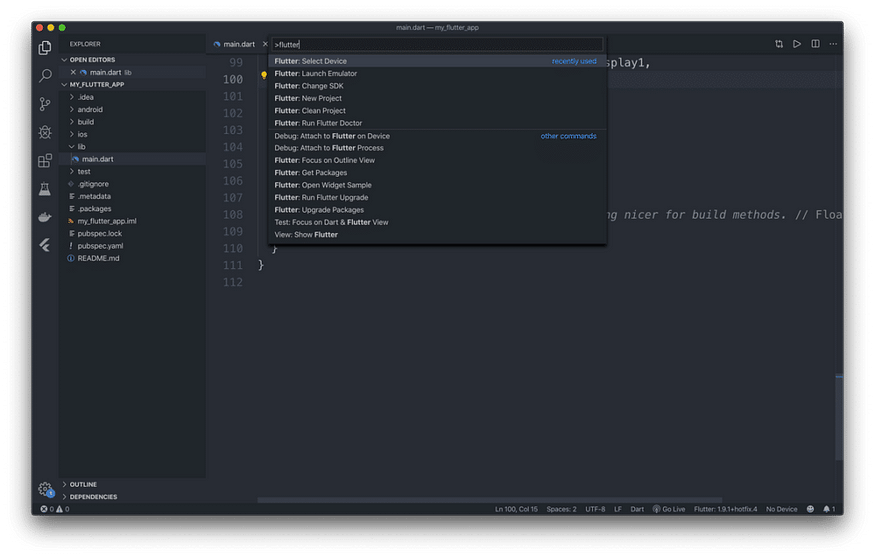
$ flutter run

In the following screenshot you can see the output which is shown on the command line after the app has been launched successfully:



Now you should be able to see the following result in the simulator:





* Here you can find commands e.g. to create a new Flutter project *Flutter: New Project*, launch the emulator with *Flutter: Launch Emulator*, or select the SDK location with *Flutter: Change SDK*.
* Once the project is created and the emulator is started you can start the Flutter project in debug mode by using the *Start Debugging (F5)* menu entry or just hitting F5 on your keyboard
* The project is then launched in the already running emulator and inside Visual Studio Code you’ll see another control bar which can be used to control the execution of the app:

