.

Flutter My Learning

**2025**

**Presented By:** MNShahCoder

Welcome to the mobile app. If you're here, you're likely looking to find random words. Random Word Generator is the perfect tool to help you do this. While this tool isn't a word creator, it is a word generator that will generate random words for a variety.

**HEADING**

**Flutter & Dart**

CONTENTS

01

**Introduction**

**Outlines:** What is CSS, Default Browser Styles, Basic CSS Syntax, 3 Ways to Add CSS, Targeting Elements, Targeting Classes and ID’s, Targeting Multiple Elements, CSS Conflicts & The Cascade.

Studying CSS will teach you to style HTML documents, understand default browser styles, and use basic syntax. You'll learn three methods to add CSS (inline, internal, external), how to target elements with selectors, classes, and IDs. Manage CSS conflicts with the cascade. This knowledge ensures and create visually appealing and consistent web pages.

02

**Inheritance & Selectors**

**Outlines:** Inheritance, Selector Specificity, Important Declaration, Descendant Selector, Child Selector, Adjacent Selector, Attribute Selector, Pseudo Selector, Hover Active & Visited Effects, First & Last Child Selector, First & Last Child of Type Selector, nth Child Selector, nth Child of Type Selector, Combining Selector, The Universal Selector.

Noted:

03

**Declarations**

**Outlines:** What is CSS, Default Browser Styles, Basic CSS Syntax, 3 Ways to Add CSS, Targeting Elements, Targeting Classes and ID’s, CSS Conflicts & The Cascade.

Studying CSS will teach you to style HTML documents, understand default browser styles, and use basic syntax. You'll learn three methods to add CSS (inline, internal, external), how to target elements with selectors, classes, and IDs. Manage CSS conflicts with the cascade. This knowledge ensures visually appealing and consistent web pages. This knowledge will enable to create visually consistent andappealing web pages.

04

**The Box Model**

**Outlines:** What is CSS, Default Browser Styles, Basic CSS Syntax, 3 Ways to Add CSS, Targeting Elements, Targeting Classes and ID’s, CSS Conflicts & The Cascade.

Studying CSS will teach you to style HTML documents, understand default browser styles, and use basic syntax. You'll learn three methods to add CSS (inline, internal, external), how to target elements with selectors, classes, and IDs. Manage CSS conflicts with the cascade. This knowledge ensures visually appealing and consistent web pages. This knowledge will enable to create visually consistent andappealing web pages.

Course1: Flutter & Dart – The Complete Guide [2025 Edt]

# Smart PhoneSection 1: Introduction

## What is Flutter?

Flutter is actually a **combination** of **two main parts**.

1 - For one, it's a UI framework, which is a fancy way of saying that Flutter is a collection of code packages and things you can use in your code to write cross-platform apps, cross-platform user interfaces, because that's one of the main selling points of Flutter.

Core packages & utility functions for writing cross-platform app code

**UI Framework**

### One Codebase, Multiple Apps

**Single Codebase**

Android

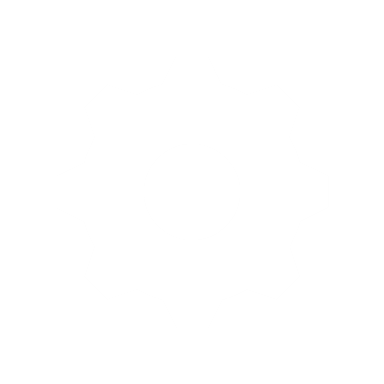
macOS

Windows

iOS

Linux

Web

****With Flutter, you can have a single codebase and yet generate apps for a broad variety of target platforms, and that's a huge advantage, because without a solution like Flutter, you would have to learn different programming languages for different platforms because, historically, there wasn't a single shared language that could have been used to target all these platforms.

CLI & software that helps with developing, testing & building cross – platform apps.

**Collection of Tools**

2 - Instead, Flutter also gives you **a collection of tools** that you use in **combination** with your code to then actually **convert** your code to code that is able to run on these different target platforms.

Because that code that you write in this single codebase does not work on these different platforms. Instead, it must be **translated to machine** **code** that does work on these different platforms.

### From Flutter Code to Platform Code

Single Codebase

Flutter translates that code to platform-specific machine code. (by Flutter Tools)

Android

iOS

Machine Code

Flutter allows you to build **multi-platform apps** based on that **one single codebase** and programming language.

With Flutter, it's just one programming language that you have to learn and use.

## Flutter is Not a Programming Language!

It’s a **Framework** for building user interfaces with **Dart**

A collection of packages & utility functions you may use in your code.

Main usage: Flutter app development.

A programming language developed by Google.

**Dart**

**Framework**

Generally speaking, things that your can use in your code.

**Example Framework**: some pre-built codes that you can add to your code that a button is displayed on the screen. That’s what the flutter framework gives you.

**Explain**: Dart is the **actual programming language** you're learning, and which cannot just be used for building mobile or cross-platform apps without Flutter, but to be fair, Flutter is the **main use case** for Dart. and you use Flutter on top of Dart, to build these cross-platform apps.

## Target Platforms

Flutter **offers** a broad variety of **possible target platforms**.

**Platforms supported by Flutter**

Initially, Flutter supported only mobile apps

Android

iOS

**Mobile Apps**

Linux

macOS

Windows

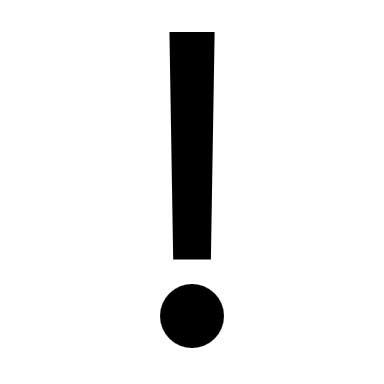
Modern browsers

**Desktop Apps**

**Web**

**Mobile Apps:** I would argue, it's still the **primary focus of Flutter** or **these platforms** are best supported by Flutter.

**Web:** Flutter for building web apps that **run** in **modern browsers** because these web apps use certain features that are **not supported** in older browsers.

Whilst you can write the code for all platforms on the same machine, you can only **test & run iOS & macOS apps on macOS machines, Windows apps on Windows machines** and **Linux apps on Linux machines!**

Android and web apps can be built on all operating systems.

## Flutter Setup

**Flutter SDK**

Always to **download** and **install** the Flutter SDK, the Flutter **software development kit**.

1- Collection of tools

2- Flutter Framework

**For managing Flutter project.**

Flutter SDK

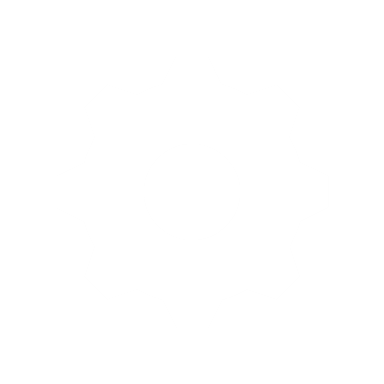
**Version control software, used internally by flutter SDK.**

Git

**Git**, which is a version control software that is generally totally **independent** from Flutter. It's a software you can use in **general as a developer** to, for **example**, create code snapshots to which you can go back if you mess something up in your code. But Flutter, these Flutter tools which you will also install, actually use this Git software **internally** to install that.

Now, with that, you would have Flutter installed, and that would be enough to write the code, but in order to also **build** the app for different platforms, and in order to **preview** the app whilst you are working on it, you also need to install some platform specific tools.

Android Studio

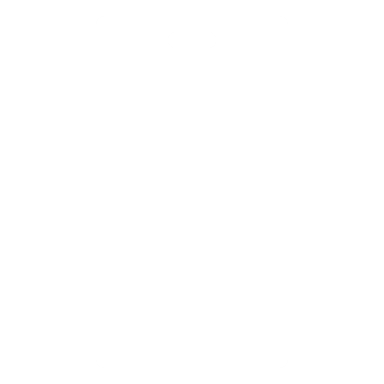
****

**Platforms Tools**

**Used by Flutter SDK & needed for Android app deployment.**

**Used by Flutter SDK & needed for iOS app deployment.**

XCode

****

iOS

Android

**Virtual Devices**

**Preview Flutter apps on virtual Android devices.**

**\**

**Preview Flutter apps on virtual Android devices.**

Using such virtual devices instead of your real phone is very **useful** because that means that you don't have to use your real phone for **testing**, and that you instead can use this virtual device which can also easily be **erased** if you need to, and which is a great tool for testing your Flutter project as you're working on the code.

### Target Platform Tools & Devices Setup

No possible

No possible

Create local Android emulator

Install SDK, command-line tools & build tools

Downloads & install Android Studio

**On Linux**

Build **Android Apps**

Build **iOS Apps**

Create local iOS simulator

Configure Xcode command-line tools

Download & install XCode

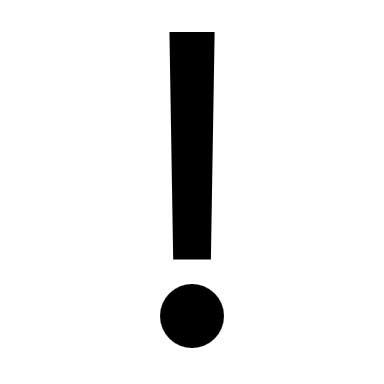
**On macOS**

**On Windows**

## Windows Setup

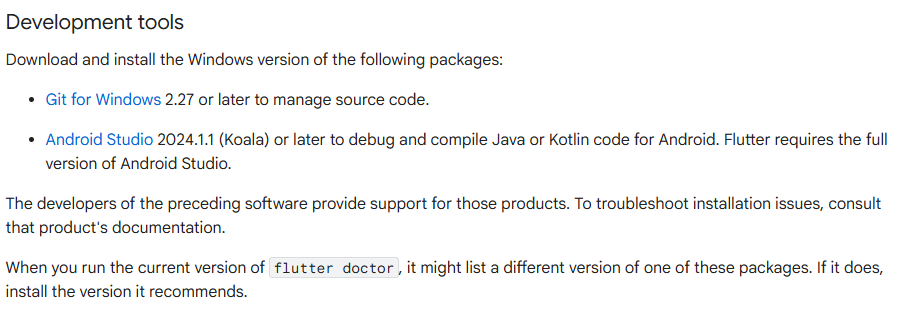
1 - This takes you to the official **Flutter documentation** and there to the **setup page** and there you can pick the **operating system** on which you are to find detailed instructions that tell you what to install on your system in order to use Flutter on it. Then, select your **target platforms**: mobile, web, or desktop. For this course, choose '**Mobile**' and refer to the installation page for setup instructions.

Link: <https://docs.flutter.dev/get-started/install/windows>

it's important to understand that on this page **you're not picking the platform for which you want to develop** but the **platform on which you develop**.

So here you **don't pick macOS** just because you want to **build** a macOS app with Flutter but instead you pick Windows because you are **on a Windows system** and you want to use Flutter on that.

2 - Now when **installing** Flutter for Windows there are some **prerequisites** for example, you must install **Git** as mentioned in the previous lecture.

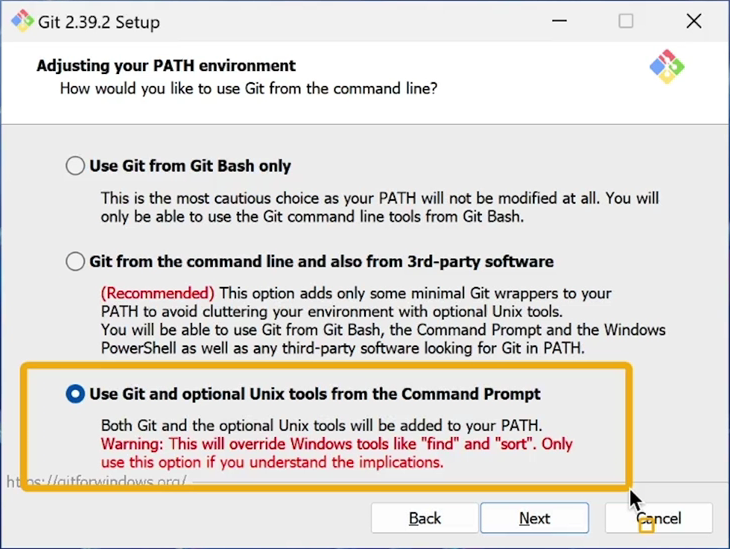


### Git Installation

Link: <https://docs.flutter.dev/get-started/install/windows/mobile#development-tools>

For that, you can simply follow this **link** on this page here which takes you to the **official Git website** and there pick the **Git version** for your **Windows system version** (64 bit or 32 bit) and open that installer, which is downloaded here.

This will now walk you through the Git installation process and in general you can **accept all the defaults** and just walk through here. But there is one setting you should override and that is that you choose use **Git and optional Unix tools from Command Prompt**.



Now with that, you can continue through the remaining steps and confirm all the defaults there, no changes needed.

### Install the Flutter SDK

You have **two options for installing the Flutter SDK**: either **Download and install it directly** to your laptop or set it up through **Visual Studio Code** (VS Code).

This **Course** it chooses to **Download and install** directly to laptop.

Link: <https://docs.flutter.dev/get-started/install/windows/mobile#install-the-flutter-sdk>

### Install the Android Studio

You have **two options for Android Studio**: either **First time using Android Studio** to new user or **Current Android Studio User** for those who already have.

This **Course** it chooses to **First time using Android Studio** to show for the viewer.  
  
Link: <https://docs.flutter.dev/get-started/install/windows/mobile#configure-android-development>

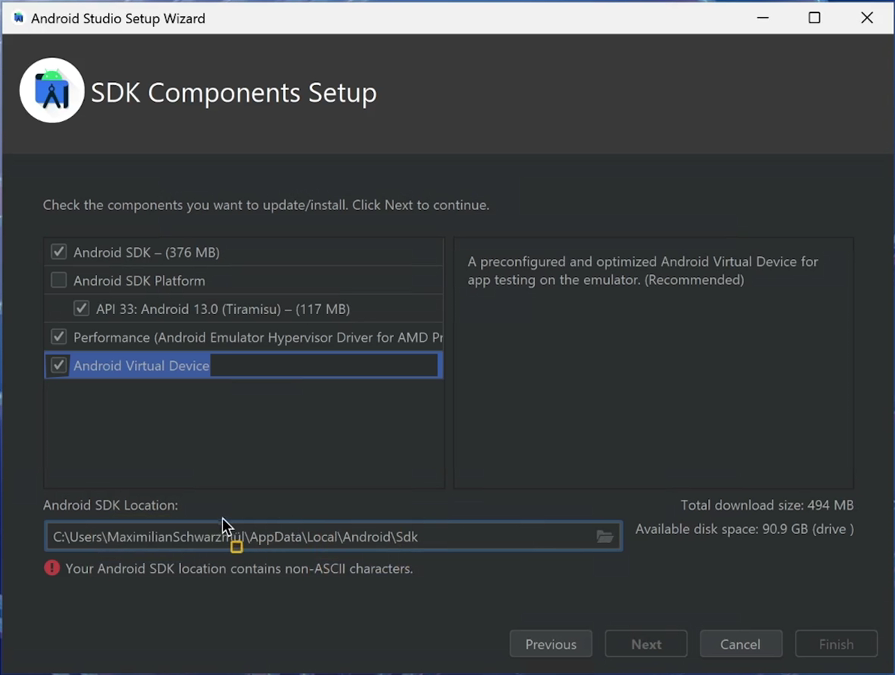
**Configure your Android device**

You have **two options for running an Android device**: either use a **Virtual Device** through the Android Emulator or connect a **Physical Device** (your own smartphone).

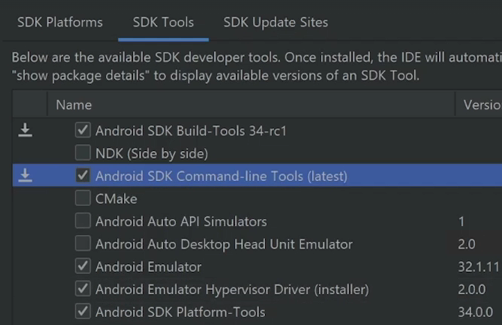
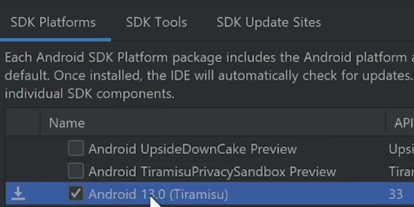
This **Course** it chooses to **Virtual Device** to show for the viewer.

Link: <https://docs.flutter.dev/get-started/install/windows/mobile#set-up-the-android-emulator>

1 - Now once this finished, you should also immediately start Android Studio. But you should make sure that you do check **Android SDK** here, that you do also pick an Android SDK platform here and **Android virtual device**.



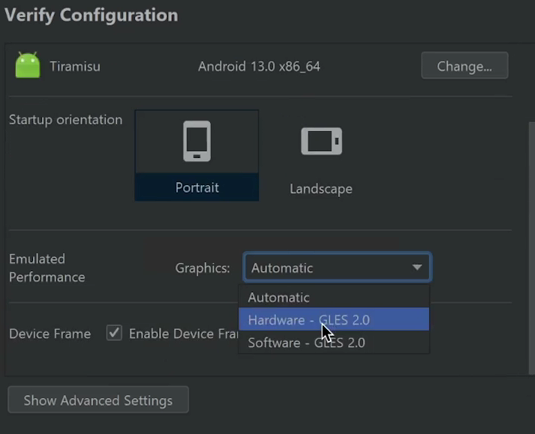
2 - So once done, you can finish this setup wizard and a new window should open where you should select **SDK manager** by clicking on **more actions** SDK manager. Here you should for one make sure that you do check that **latest stable Android version** here under **SDK platforms** andunder **SDK tools**, you should select the **SDK build tools** and make sure that you also got the **SDK platform tools** selected here.In addition, the **Android SDK command line tools** should also be selected.



3 - Again, go to **more actions** here and go to **virtual device manager** and here, click on **Create device**.

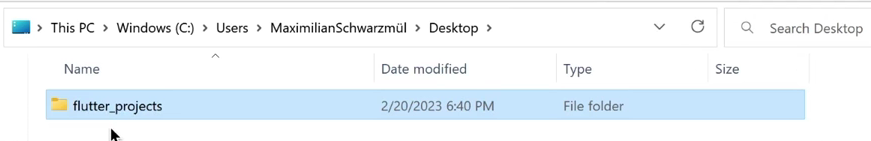
This wizard allows you **to create a new virtual device blueprint** and the idea behind the virtual devices, simply is that you have an Android device emulator running on your system **to** **conveniently preview your Flutter app** whilst you are working on the code.

So here you can pick any **template / types device** of your choice like **pixel 6** in my case, click next, **download** the **latest Android version** here, which again can take a short while. On this page under emulated performance pick **hardware** if possible and then click finish with that.



## Project Creation & Setting Up a Code Editor for Flutter Development

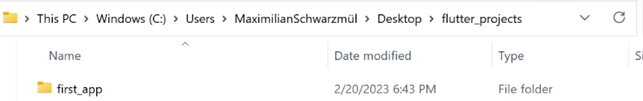
1 - First, create a **new folder** anywhere on your system. The **folder name** should **not contain spaces** or **dashes**; use **underscores** instead. Once the folder is created, you can use it to store your Flutter projects.



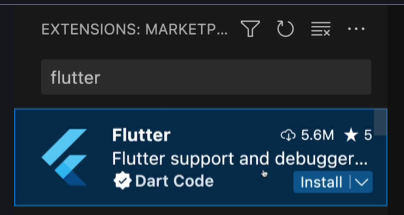
2 - To do that, simply open in your **terminal**—**Command Prompt** or **PowerShell** on Windows. You can right-click the folder and select **'Open in Terminal'** to navigate to its location. You can create a new Flutter project by running **Flutter create**, and then choosing any name of your choice. For example, **first\_app**.



This command will now **create a new project** folder inside of that folder that you created, where a **new Flutter app**, a **new Flutter project** will be created in.



3 – **Download** Visual Studio Code (VS Code) and open the folder named **first\_app.** Then, **install** the **Flutter extension** — an official extension provided by the Flutter team. This extension is essential for Flutter development and is a **must-have** **when using VS Code.**



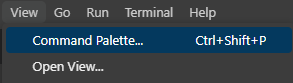
## Running a First Flutter App

This **main.dart** file here of course already has some code in it, soon we can see something on the screen, all this code won't tell you much though. We can use this **starting code**, this demo code which I provided to you here to **see a first app running** on an **emulator**.

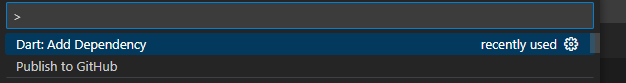
### Running Emulator in VS

A simple way of running such an emulator is to use a VS Code and then they are under view.

1. Choose the **Command Palette** or use the **shortcut (Crtl + Shift + P)** which you see here.



1. Which will open this **Command Windows** here where you can type **commands** and **execute** them from inside VS Code.



1. If you type Flutter here, you'll see some Flutter related commands. Then start typing

**Flutter: Launch Emulator** and click **Enter**.

1. If you choose that a new window should open up that allows you to **pick an emulator**. And here I'll use that **Android emulator** which I created before **Pixel 6 API 33**. Now with such a virtual device up and running, we can run our Flutter app on it.

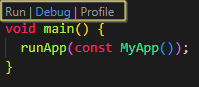
### Add Status Bar

First of all, I recommend that you go to **view** and then there under **appearance**, you make sure that you show the **status bar** which is this bar at the bottom, there you should see the device which will currently be used for running your Flutter app here. And if you click on this device, you could also change it and for example test it as a **macOS** or **Windows app** or anything like this.

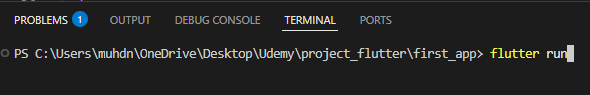


### Ways to Run the Code

1. Going to the **main.dart** file and you should see a couple of options above this main thing here. It has a **Run** option and **press** this option her.



1. Now alternatively, you can also use the **terminal**, which you can also open through VS Code and then there choose **new terminal** and they’re run **Flutter run**.



1. You can go to the **menu bar** and there under **run** you find the **start debugging** and **run without debugging** options. In most cases in this course, I'll used this run without debugging option.



1. Go to **main.dart**. In the top right corner, next to the file name, you can see the **Run symbo**l; click to run.



### Debug Console

This window here also opens up this debug console in which you could see **error messages** if something goes wrong, you can also close this window though the build process will still continue.

### Control Panel Apps

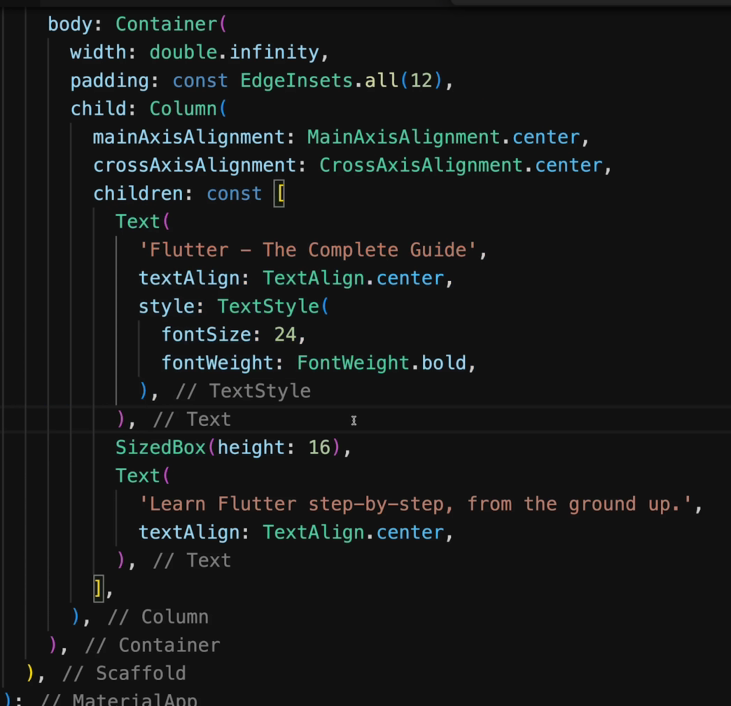


This can be used to control the running app. You can, for example, **stop** it by pressing this **stop button** here. You can **restart** it to make sure that all the **changes** you might have made in code are picked up by **hot reload button**. In that case, you can press this **flash** **button** here to **force** Flutter to take another look at your code and apply these changes.

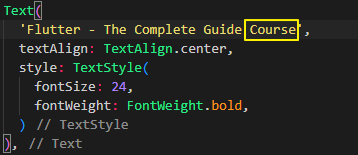
These features simply exist to make the overall Flutter app development process as convenient as possible since **you don't have to restart the entire emulator** or **wait for the app to be rebuilt every time you make a tiny change**. Instead, as you saw, when you make a change, it's typically reflected in the running app instantly.

### Change Code to main.dart

Let's also make our first little code change to see how that could be done and how those changes would be reflected on that virtual device. Insert this to **stateless widget** and try to **run** it at main.



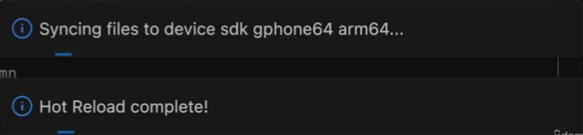
Let's say we want to change this code and we want to rename this to **Flutter the complete guide course**. We can simply make that change and if you then save that code.



At the moment it's **unsaved**, as you can tell in VS Code by this **dot** here. That's showing up next to the file name in this tab.



But once you **save** that change, for example, by pressing **control S** or **command S** on macOS, you will see that we had these **little Windows open** up at the **bottom** here which then tell us that the **Flutter extension** that we installed here in VS Code **noticed those changes** and **automatically updated** the app running on the virtual device.



## Understanding Material Design

**MaterialApp()** Widget:

The name **MaterialApp** includes the word **Material**, which is important because Flutter apps by **default** use **Material design**, which is **Google's flexible design system**.

Now, you can learn more about Material design on **Material.io** which is the official webpage of the Material design system and language.

And in the end, this Material design thing is simply **a set of** **suggestions**, **rules**, **and** **guidelines** established by Google that help you build beautiful user interfaces.

**About Material Design**

**Google’s flexible design system**

A set of suggestions, rules & guidelines that help you build beautiful user interfaces.

Using **MaterialApp** widget you get a lot of **base** **styles** and **rules** that ensure that your mobile user interfaces that are built with Flutter will always look nice.

## About This Course

**Dart & Flutter Fundamentals**

Base syntax, core features & foundational concepts needed to build mobile app user interfaces

**Advanced Features**

Handling user interactions, customizing styles, building multi-screen apps.

**Beyond the Basics**

Animating apps, connecting a backend, using native device features (e.g., camera) & more.

## Course Resources (Code Snapshots, Community & Slides)

This course also comes with free access to our “**Academind Community**” on Discord: <https://academind.com/community/>

---

There, you can find **more than 200.000** **like-minded people**, **discuss** issues, **help each other**, share progress, successes and ideas and simply **have a good time**!

I believe that you learn the most if you **don’t learn alone**but find learning partners and other people with similar interests. Our community is a great place for this - it’s the perfect complimentary resource for this course.

**Joining it is of course free and 100% optional.**

---

You find course resources (**code snapshots** & **slides**) on **GitHub**, in this repository: <https://github.com/academind/flutter-complete-guide-course-resources>

The folders in the **Code** folder map to the different course sections.

The folders inside those subfolders (e.g., **Code/02 Flutter & Dart Basics/01 Starting Setup**) represent individual code snapshots - e.g., the starting project code for a course section.

You also find the **course slides** in the above repository - or you use this [direct link](https://github.com/academind/flutter-complete-guide-course-resources/blob/main/Slides/course-slides.pdf).

# Section 2: Flutter & Dart Basics I – Getting a Solid Foundation [ROLL DICE APP]

## Module Introduction

In this course section, we're going to build a **first basic app**, an app which allows us to **roll the dice**, and **generate random values** that are then reflected by an appropriate dice image in our app.

Basically, all Flutter apps you're about to build because this course section is about

* Explore Core Flutter & Dart Syntax
* Understanding & Writing Flutter and Dart Code
* Working with Flutter Widgets

## Analyzing A New Flutter Project