

DART AND FLUTTER

Introduction

The purpose of this chapter is to give the reader a quick introduction to the world of Mobile Application development, and to introduce him or her the problem of cross-platform development, and how it was approached by different companies.

A mobile app or mobile application is a computer program or software application designed to run on a mobile device such as a phone/tablet or watch. Writing mobile apps sounds easy but is complicated by the number of platforms that are available. Your app could run on an iPhone, it could run on an iPad, it could run on an Android Phone etc. Also remember that these platforms could change quickly as new devices appear on the market.

Before Cross-Platform Mobile Application Development

In the past, in order to produce performant applications, developers had to write the application code specifically for each platform. There would often be one codebase (and developers) for iOS (iPhone) and another codebase (and developers) for Android. For iOS, Objective-C and Swift are the preferred programming languages. For Android, Java is the preferred language.

- You had to keep two sets of code in sync.
 - If you change the iPhone code, you should change the Android code to match.
- You had to have developers with multiple skillsets. Expensive.
- Sometimes the app for one platform would look very different from the other platforms.

Early Cross-Platform Development Tools

Anyway, Silicon Valley soon realized what a problem this was and set to work on developing tools for cross-platform mobile application development. They quickly split into two groups of development tools: those that used native libraries and those that didn't.



Development Tools That Used Native Libraries

These tools created a 'Unified' API on top of the native SDK supplied by Apple and Google. The problem with these types of applications is that the 'Unified API' does not cover 100% and leaves the developers with many burdens, such as having to still write a large chunk of platform-specific code. Many of these development tools, for example Xamarin, Appcelerator, Native script are still around.



Development Tools That Didn't Use Native Libraries

These tools took a different approach. Most of these attempted to bypass the SDK approach and write code that runs on the platform's browser. This had the advantage of being able to use many of the HTML5 and JavaScript capabilities already built-in. The app would run in a 'web view'. A "WebView" is a browser bundled inside of a mobile application producing what is called a hybrid app. Using a WebView allows mobile apps to be built using Web technologies (HTML, JavaScript, CSS, etc.) but still package it as a native app and put it in the app store. The problem with these types of applications is speed. They are not running natively in compiled machine code, they are running on a hidden web browser.

Many of these development tools, for example Cordova, PhoneGap are still around.

Modern Cross-Platform Development Tools

More recently, two main rivals have emerged and look to be leading the field of mobile app development tools: Facebook React Native and Google Flutter.

React Native

React.JS is an excellent JavaScript framework that has been popular for years and works with both mobile and non-mobile websites equally well. Developers write user interfaces with Component objects, like lego blocks. These Components can contain code so that they can react to the user's input and produce an interactive user interface. React Native is like React, but it uses native components instead of web components as building blocks.

How Does It Work?

React Native runs in two parts.

- The UI. It displays the ui and receives user input.
- The JavaScript engine. It interprets and executes the JavaScript application code.

The two parts communicate with a bridge.

Conclusion

React Native is an excellent framework. It has the great advantage of being the more established player because it has been out since 2015. There are also a lot of React developers out there who can quickly cross-train to use React Native rather than React JS. React Native is also a very productive tool because it has many ready-to-use components.

However, at the end of the day it runs an efficient, native UI with interpreted JavaScript, communicating through a bridge. This is not the optimum solution for performance.

Companies uses React native :

1. Bloomberg
2. Facebook
3. Uber Eats
4. Discord
5. Instagram
6. Skype
7. Pinterest
8. Salesforce
9. Baidu
10. Walmart
11. Wix
12. Shopify
13. Words with Friends
14. Vogue
15. NerdWallet

Google Flutter

Google Flutter has only been available since 2017 but it is making waves because it takes a different approach to cross-platform mobile app development. Google is currently working on the successor to its Android operating system called Fuchsia and it is writing it using Flutter. So, Flutter is very important to Google.

You write user interfaces using Google Flutter user interface widgets, not the native iOS or Android ui widgets shipped with their retrospective SDKs. A Flutter app made using Flutter widgets will look exactly the same on iOS as it does on Android. Flutter comes with many widgets, including those that mimic Google's Material look & feel and those that mimic Apples iOS look & feel.

Google Flutter uses its own high-performance rendering engine to draw these widgets and they have been designed to work on all mobile platforms. Also, these widgets are extendable.

You write the application code in Google's Dart language and it is compiled ahead-of-time into machine-code for native-like performance, thus offering a performance advantage over React Native.

There is no bridge between the user interface and the application code.

The only downside that is currently obvious is that developers will have to learn Dart, rather than reuse their existing JavaScript expertise.

Conclusion

If you want to write cross-platform mobile web apps that are performant then Google Flutter appears to be the best choice at the moment. However, things move quickly and that may not be for long!

Companies using flutter :

1. Xianyu app by Alibaba ([App on App Store](#), [App on Google Play](#)) – Alibaba is one of the biggest e-commerce companies in the world
2. Hamilton app ([App on App Store](#), [App on Google Play](#), [Website](#)) – official app for the Broadway Musical
3. Google Ads app ([App on App Store](#), [App on Google Play](#))
4. eBay Motors app ([App on App Store](#), [App on Google Play](#)) – it's a powerful tool for browsing, buying, and selling vehicles directly from consumers' phones
5. Stadia ([App on App Store](#), [App on Google Play](#)) – Google's gaming platform, users can play games across laptops, desktops, and mobile devices
6. Groupon ([App on App Store](#), [App on Google Play](#)). Groupon uses Flutter to help hundreds of thousands of merchants track campaign performance, manage customer satisfaction, and get efficient support.
7. Baidu Tieba ([App on App Store](#)) – the largest Chinese communication platform hosted by Chinese search engine company Baidu.
8. Phillips Hue ([App on App Store](#), [App on Google Play](#)) – Flutter brought intuitive controls to Philips Hue apps, allowing users to set the mood and sync their smart lights to their media.
9. Topline app by Abbey Road Studios ([App on App Store](#), [App on Google Play](#))
10. Reflectly ([App on App Store](#), [App on Google Play](#), [Website](#))

INSTALLATION

There are many complex IDE(s) that support Dart by plugins, such as Android Studio, IntelliJ IDEA, Visual Studio Code, etc. It depends on your purpose to choose the right IDE. If you just want to learn the Dart language, Visual Studio Code is a good choice, because it is a powerful, lightweight and free IDE.

By default, Visual Studio Code supports popular languages (or Microsoft language) such as C/C++, C#, F#, Visual Basic, HTML, CSS, JavaScript, etc, but not including Dart. Therefore, to program Dart in Visual Studio Code, you have to install Dart extension.

Dart is a client-optimized programming language that supports various areas like multi-platform application development, formatting, analyzing, and testing code. It is a programming language that forms the basic foundation of Flutter.

Flutter is an open-source UI – Software development kit that helps develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web in a single code base.

Visual Studio Code is a code editor developed by Microsoft for Windows, Mac, and Linux with features like code debugging, syntax highlighting, code completion, code refactoring, etc.

[Dart SDK](#)

Installation:

Follow the below steps to install the dart SDK:

Step 1: Open “[Get the Dart SDK](#)” website or Open “[Get the Flutter SDK](#)” website.

Note 1: “Dart SDK” OR “Flutter SDK” in following steps.

Note 2: Dart SDK is part of Flutter SDK. So, if you are installing “Flutter SDK”, then no need to install “Dart SDK” separately.

Note 3: Installation for both are exactly same.

Step 2: Scroll down to the bottom of the website and click “[Downloading the SDK as a zip file](#)”.

Step 3: Based on your device's architecture, Click the *"Dark SDK"* for Windows or Mac or Linux, accordingly.

Step 4: Once the download is complete, go to the *"Downloads"* folder and unzip the newly downloaded Dart-SDK zip file.

Step 5: For unzipping the file, right-click on the file and click *"Extract Here"*.

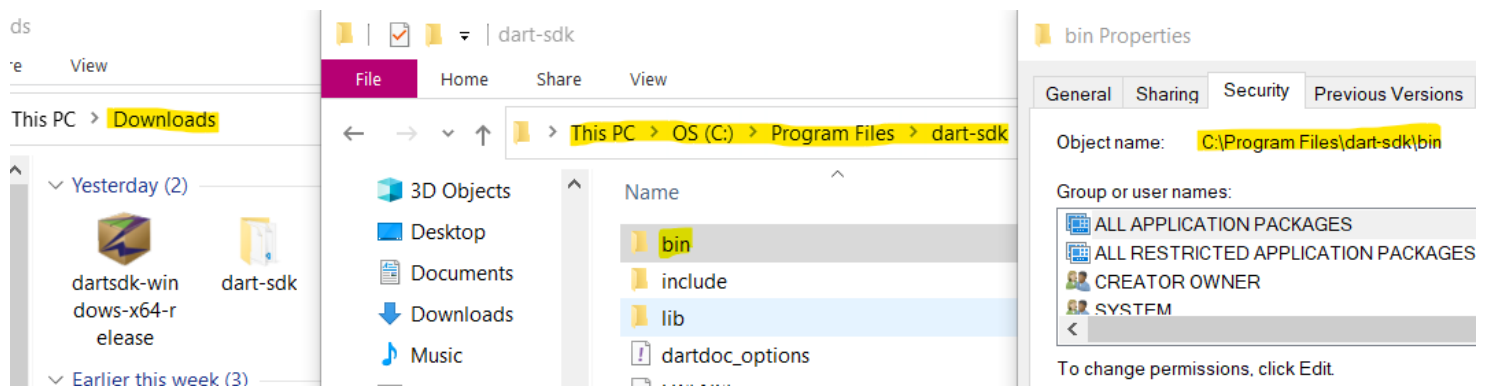
Step 6: Now, a new *"dart-sdk"* folder will be created in the *"Downloads"* folder.

Step 7: Copy this file and paste it in the *"Program Files"* folder in the *"C"* drive.

Step 8: Open *"dart-sdk"* folder and you will find a folder named *"bin"*.

Step 9: Now, right-click on the *"bin"* folder and choose *"Properties"*.

Step 10: In the *"bin Properties"* window, go to *"Security"* and copy the *"Object Name"*, also known as the address of the file, and press *"Ok"*.



Installation of Dart SDK

Setup:

Step 11: Open *"This PC"* folder, right-click inside this folder, and choose *"Properties"*.

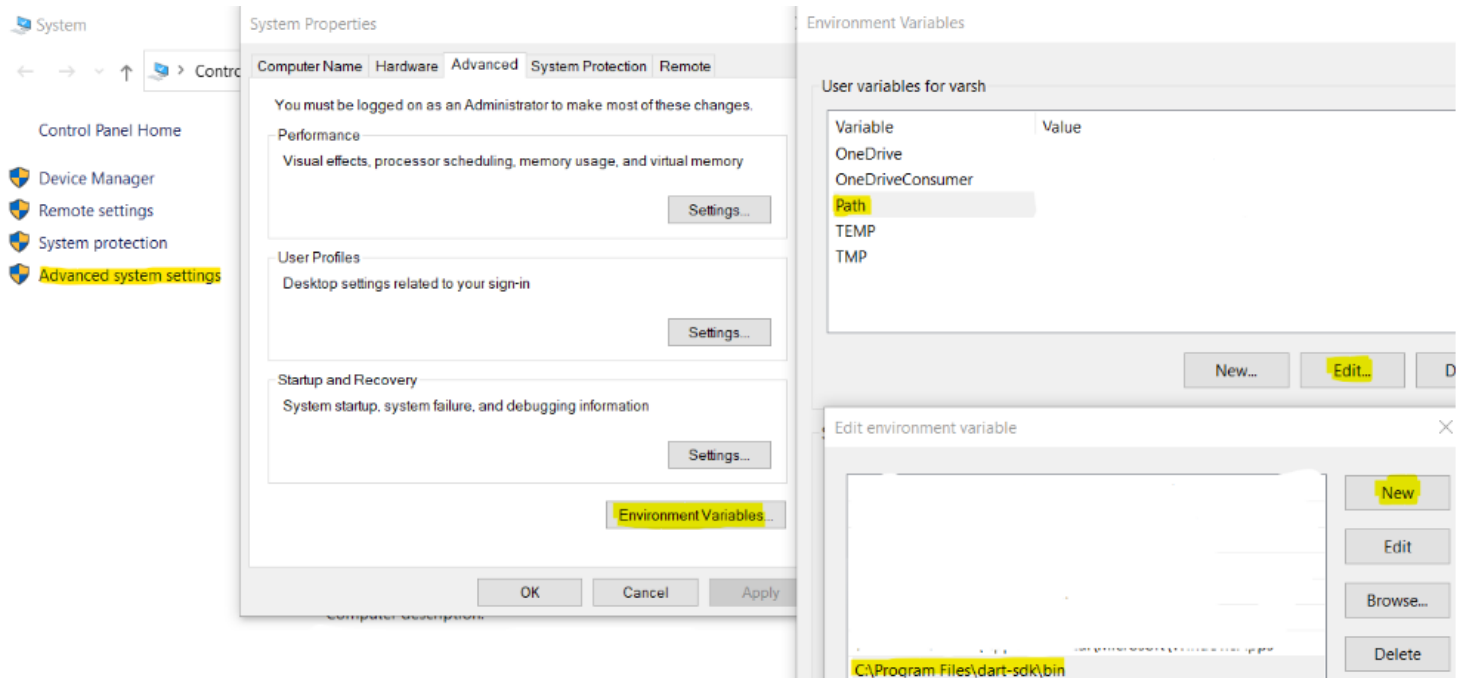
Step 12: In the *"System"* window, select *"Advanced system settings"* that will be available on the left-hand side of the window.

Step 13: In the *"Advanced"* tab, press the *"Environment Variables..."* button.

Step 14: In the *"Environment Variables"* window, under *"User Variables for..."* check for the *"Path"* variable.

Step 15: If it's available, then press the *"Edit"* button, press *"New"* button in the *"Edit environment variable"* window, add the copied address to it and save the made changes.

Step 16: If it's unavailable, then create one by pressing the *"New"* button and entering the *"Variable Name"* as *"Path"* and *"Variable Value"* as the copied address, and press *"Ok"*.



Setting Environment variables for Dart SDK

Step 17: Now open the Command Prompt (cmd) and type “*dart*” / “flutter doctor”

Step 18: The output will be as follows.

- Dart

```
|| The Dart tool uses Google Analytics to anonymously report feature usage ||
|| statistics and to send basic crash reports. This data is used to help ||
|| improve the Dart platform and tools over time. ||
```

```
|| To disable reporting of anonymous analytics, run: ||
```

```
|| dart --disable-analytics ||
```

A command-line utility **for** Dart development.

Usage: dart [<vm-flags>] <command|dart-file> [<arguments>]

Global options:

-h, --help	Print this usage information.
-v, --verbose	Show additional command output.
--version	Print the Dart SDK version.
--enable-analytics	Enable anonymous analytics.
--disable-analytics	Disable anonymous analytics.

Available commands:

analyze	Analyze the project's Dart code.
compile	Compile Dart to various formats.
create	Create a new project.
fix	Apply automated fixes to Dart source code.
format	Idiomatically format Dart source code.
migrate	Perform a null safety migration on a project or package.
pub	Work with packages.
run	Run a Dart program.
test	Run tests in this package.

Run **"dart help <command>"** **for** more information about a command.

See <https://dart.dev/tools/dart-tool> for detailed documentation.

Step 19: If your output is just like above then, it signifies that dart has been installed successfully.

Visual Studio Code

Installation:

Follow the below steps to set up dart in VS Code:

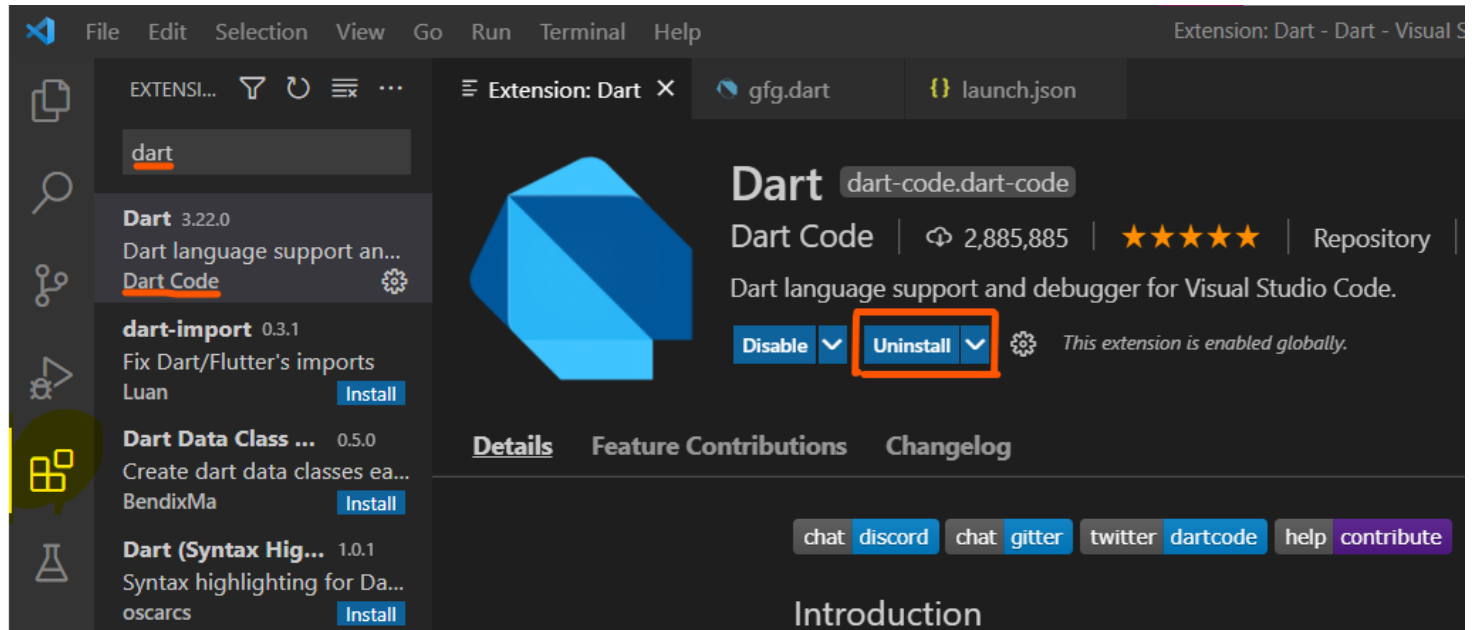
Step 1: Open Visual Studio Code in your device.

Note: If unavailable then, open “[Download Visual Studio Code](#)” and download the application based on your device’s architecture and configurations.

Step 2: Press “*Ctrl + B*” and select “*Extensions*” or directly Press “*Ctrl + Shift + X*”.

Step 3: In the search bar, type “*dart*” and open the extension named “*Dart*” by “*Dart Code*” in the list. Press the “*Install*” button.

Step 4: In the search bar, type “*flutter*” and open the extension named “*Flutter*” by “*Dart Code*” in the list. Press the “*Install*” button.



Installing Dart in Visual Studio Code

Step 4: Once the installation is complete, click “*File*” and “*New File*”.

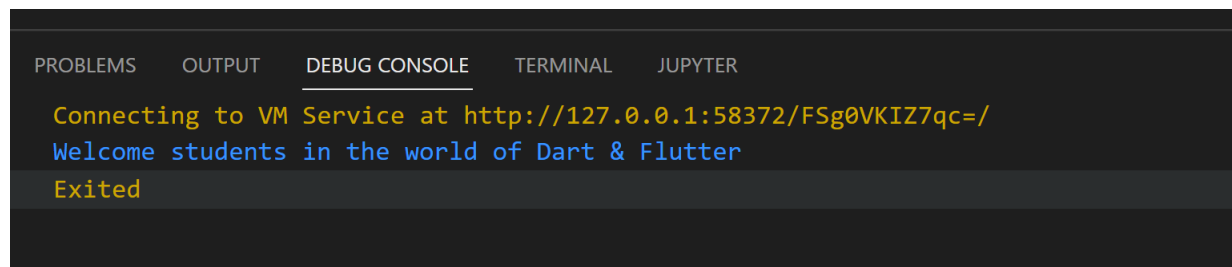
Step 5: Again click “*File*”, “*Save As*” and name the file with a “*.dart*” extension like “*tutorial_01_01.dart*”. This file should be saved under a folder, but not as a loose file. So, if needed to create a new folder named “*Dart*” and save this new file under this folder. Now, let’s code our first Dart program.



Step 6: The above code print the string “*Welcome students in the world of Dart & Flutter*”. Type or copy the above code in the coding area.

Step 7: Save the file by pressing “*Ctrl + S*” or by selecting “*File*” and “*Save*”.

Step 8: Now for running the code, press “*Ctrl + Shift + D*” and press “*Run and Debug*” button or open “*Run*” and select “*Start Debugging*”.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

Connecting to VM Service at http://127.0.0.1:58372/FSg0VKIZ7qc=/
Welcome students in the world of Dart & Flutter
Exited
```

Note: perform following steps only if you don’t get output above.

Step 9: Now a new file named “*launch.json*” will be created and it opens automatically in VS Code.

- Dart

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit:
    https://go.microsoft.com/fwlink/?linkid=830387

    "version": "0.2.0",
    "configurations": [
        {
            "name": "Dart & Flutter",
            "request": "launch",
            "type": "dart",
            "program" : "tutorial_01_01.dart"
        }
    ]
}
```

Step 10: In “*launch.json*” file, add “*program value*” inside the curly brackets ({}), like for example {“*program*”: “*tutorial_01_01.dart*”} as shown above and save the file.

Note: If needed add a “,” (comma), to the previous value of “*program*” in the code of “*launch.json*” file.

Step 11: Now, press *“Run”* and then click *“Run without Debugging”*. The output of the dart file will be displayed in the *“Debug Console”*.

Step 12: Now for running the code in a terminal, press *“Terminal”* and click *“New terminal”*.

Step 13: Now in the terminal, type *“dart file_name”* like *“dart tutorial_01_01.dart”* and press *“Enter”*.

Step 14: The output of the file will get displayed in the terminal.