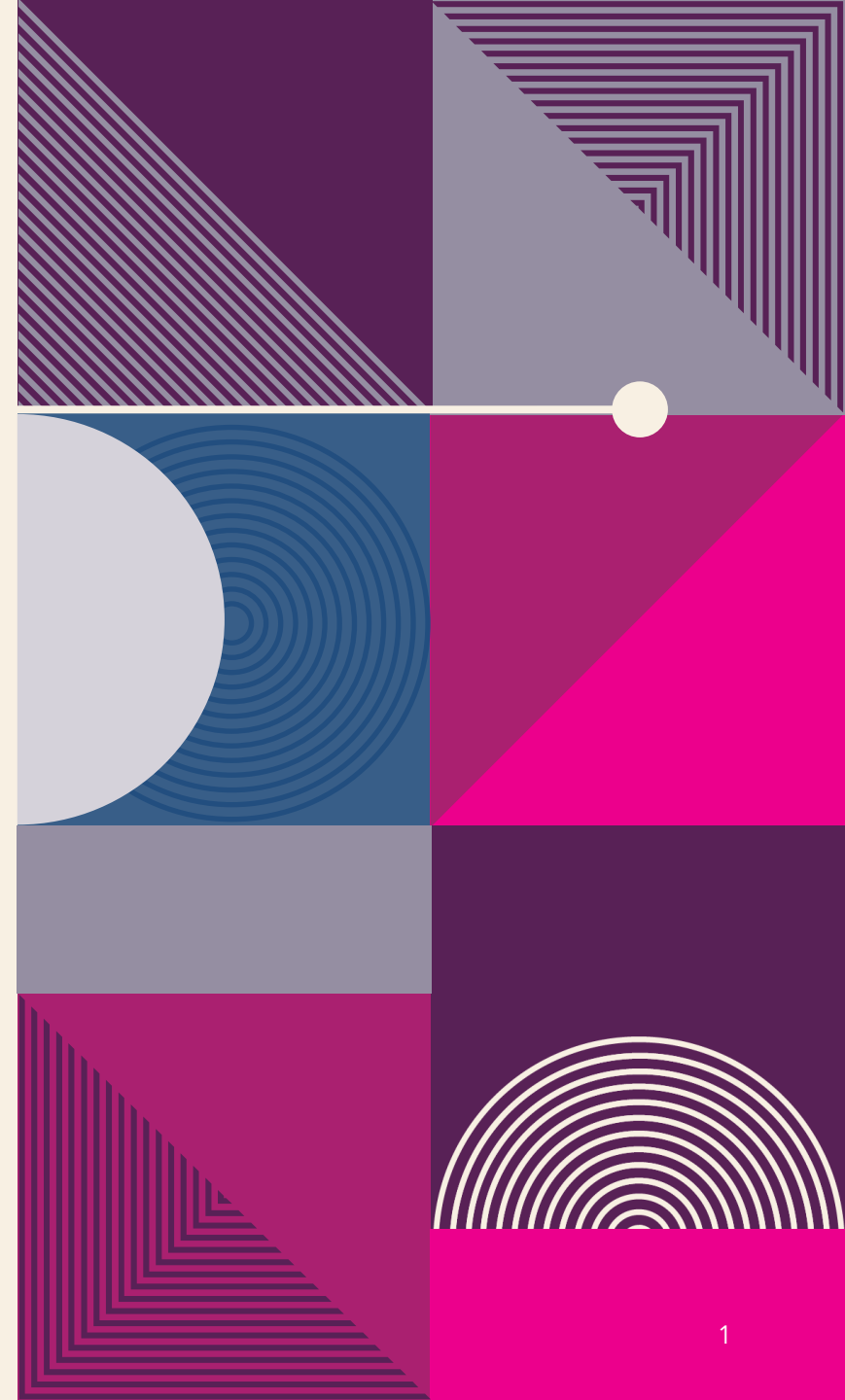


Mobile App Development



- **Native App**

The term native app development refers to building a mobile app exclusively for a single platform. The app is built with programming languages and tools that are specific to a single platform. For example, you can develop a native Android app with Java or Kotlin and choose Swift and Objective-C for iOS apps.

- **Cross-Platform App**

A cross-platform app is coded once and is compatible to run on multiple platforms. In other words, there is a single, unified code that works for both Android and iOS apps.





Flutter Framework

- Flutter is Google's Framework for building mobile applications for iOS and Android (Cross-platform).
- Although it's relatively new, Flutter has experienced a great deal of experimentation and evolution over the years.
- It was called Sky, at its first appearance at the Dart Developer Summit 2015 presented by Eric Seidel.
- Presented as Flutter in 2016, and with its first alpha release in May 2017, it was already building for iOS and Android systems. Then it started to mature and community adoption began to grow. It evolved from community feedback to its first stable release on December 5, 2018, at an annual developers' event in London, Google announced the release of Flutter 1.0.

Cross-platform development comes in three general flavours

	Some technologies	Cons	Pros
Progressive Web Apps (PWA)	HTML/CSS, React, Angular, Vue	Not a real app. Runs in a web browser. Not available in app stores. Hard to create a desktop shortcut. Cannot access many of the device's resources like accelerometer, compass, proximity sensor, Bluetooth, NFC, and more	Easy to write
Hybrid	PhoneGap, Cordova, Sencha, Ionic	Runs in a WebView so it can be slow. Nearly impossible to share code with the web app	Easier for web devs to learn because it uses HTML and JavaScript as its language and structure
Compile-to-native solutions	React Native, NativeScript, Flutter, Xamarin	Learning a framework may be difficult. Mastering the toolchain definitely is	Real apps that can be found in the stores and run fast

Compile-to-native cross-platform frameworks

	 Xamarin	 NativeScript	 React Native	 Flutter
Year introduced	2011	2014	2015	2018
Backed by	Microsoft	Telerik	Facebook	Google
Presentation language	XAML and/or xamarin.forms	Proprietary but looks like XML	Proprietary but looks like JSX	Dart
Procedural language	C#	JavaScript	JavaScript	Dart

Differences between existing frameworks

- There are a large number of high-quality and well-accepted frameworks and technologies.
- Some of them are as follows: – Xamarin, React Native, Ionic, Cordova
- So, you might think it's hard for a new framework to find its place on an already full field, but it's not. Flutter has benefits that make space for itself, not necessarily by overcoming the other frameworks, but by already being at least on the same level as native frameworks:
 1. High performance
 2. Full control of the user interface
 3. Dart language
 4. Being backed by Google
 5. Open source framework
 6. Developer resources and tooling

Flutter system requirements on Windows

- The build process consumes lots of resources especially CPU. If your machine is too old, it will be too hot and you may have to pay more time than usual as well as suffering stress while developing apps with Flutter.
- OS: Windows 10 64-bit or higher
- CPU: Intel Core i5-8400
- Memory: 8 GB RAM
- Free storage: 5 GB SSD
- Tools: Windows PowerShell 5.0+, Git 2.x

Flutter system requirements on Mac

- OS: macOS 10.10 or newer
- CPU: Intel Core i3-8100 or equivalent
- Memory: 8 GB RAM
- Free storage: 5 GB SSD
- Pre-installed tools: bash, curl, git 2.x, mkdir, rm, unzip, which, zip

Flutter system requirements on Linux

- GNOME or KDE desktop - Tested on gLinux based on Debian.
- 64-bit distribution capable of running 32-bit applications
- GNU C Library (glibc) 2.19 or later
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
- 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Flutter system requirements on Chrom OS

- 8 GB RAM or more recommended
- 4 GB of available disk space minimum
- 1280 x 800 minimum screen resolution
- Intel i5 or higher (U series or higher) recommended

IDE For Flutter

- Xcode 12.5.1 or later. Xcode is iOS's main development tool, so you need it to build your Flutter app for ios.
- Android Studio, Android Studio is iOS's main development tool, so you need it to build your Flutter app for android os.
- You have the option of using Visual Studio Code for your Flutter development environment instead of Android Studio. You'll still need to install Android Studio to have access to the Android SDK and an Android emulator

Dartpad and Dart Language

- Dartpad is a browser-based text editor that can execute Dart code, created by the Dart team. It's a "playground" or "scratchpad" of sorts, useful for testing small pieces of code. If you're just starting out with Dart, you can use Dartpad to get your feet wet without having to install the Dart SDK and IDE plugins.

- DartPad is an open source tool that lets you play with the Dart language in any modern browser. Many pages — especially codelabs — have embedded DartPads.

<https://dartpad.dev/>

Flutter Windows installation

Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK: [flutter_windows_3.22.0-0.3.pre](#)
2. Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK (for example, C:\Users\<your-user-name>\Documents).
3. Warning: Do not install Flutter in a directory like C:\Program Files\ that requires elevated privileges.

Flutter Windows installation

- Update your path
- If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the PATH environment variable:
- From the Start search bar, enter 'env' and select Edit environment variables for your account.
- Under User variables check if there is an entry called Path:
 - If the entry exists, append the full path to flutter\bin using ; as a separator from existing values.
 - If the entry doesn't exist, create a new user variable named Path with the full path to flutter\bin as its value.
- You have to close and reopen any existing console windows for these changes to take effect

Flutter Windows installation

- Run flutter doctor
- From a console window that has the Flutter directory in the path, run the following command to see if there are any platform dependencies you need to complete the setup:
- `C:\src\flutter>flutter doctor`
- This command checks your environment and displays a report of the status of your Flutter installation. Check the output carefully for other software you might need to install or further tasks to perform (shown in boldtext).
- For example:
- [-] Android toolchain -develop for Android devices • Android SDK at D:\Android\sdk X Android SDK is missing command line tools; download from <https://goo.gl/XxQghQ>
- Try re-installing or updating your Android SDK, visit <https://docs.flutter.dev/setup/#android-setup> for detailed instructions

IDEs for Flutter app development

- There are many IDEs available for cross-platform mobile application development such as Android Studio, Visual Studio Code, Xcode, and IntelliJ.

On which Android Studio and Visual Studio Code are the most common IDEs for Flutter app development.

Install Android Studio

- Download and install [Android Studio](#)
- Start Android Studio and go through the 'Android Studio Setup Wizard'. This installs the latest Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, which are required by Flutter when developing for Android.
- Run flutter doctor to confirm that Flutter has located your installation of Android Studio. If Flutter cannot locate it, run `flutter config --android-studio-dir <directory>` to set the directory that Android Studio is installed to.
- Install plugins for your code editor:
 - [Flutter](#) and [Dart](#) plugins installed for Android Studio.
 - [Flutter](#) extension installed for Visual Studio Code.

Set up your Android device

- To prepare to run and test your Flutter app on an Android device, you need an Android device running Android 4.1 (API level 16) or higher.
- Enable Developer options and USB debugging on your device

Detailed instructions are available in the [Android documentation](#).

- Windows -only: Install the [Google USB Driver](#).
- Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access your device.
- In the terminal, run the flutter devices command to verify that Flutter recognizes your connected Android device. By default, Flutter uses the version of the Android SDK where your **Android Debug Bridge (adb)** adb tool is based. If you want Flutter to use a different installation of the Android SDK, you must set the ANDROID_SDK_ROOT environment variable to that installation directory.

Set up the Android emulator

- To prepare to run and test your Flutter app on the Android emulator, follow these steps:
- Enable [VM acceleration](#) on your machine.
- Launch Android Studio, click the AVD Manager icon, and select Create Virtual Device...
 - In older versions of Android Studio, you should instead launch Android Studio > Tools > Android > AVD Manager and select Create Virtual Device....
(The Android submenu is only present when inside an Android project.)
 - If you do not have a project open, you can choose Configure > AVD Manager and select Create Virtual Device...
- Choose a device definition and select Next.
- Select one or more system images for the Android versions you want to emulate and select Next. An x86 or x86_64 image is recommended.
- Under Emulated Performance, select Hardware - GLES 2.0 to enable [hardware acceleration](#)
- Verify the AVD configuration is correct and select Finish.