What program languages should I use? (Java*, C#, C++, or HTML5)

Do you have a breakout idea for an Android* app but don’t know how to fully realize it through the available programming tools and languages? This article gives you an overview of the key programming languages used in Android development.

Android’s rich and strong capabilities let you have a choice: what to develop and how to do it. There are a growing number of instruments including program languages, IDEs, and frameworks that make the development process more and more flexible, easier, and efficient. This diversity provides a low entry barrier for programmers to become a part of the Android industry and attracts new resources.

The most popular languages are Java, C#, C++, and HTML5. Each of them has its own history, purposes, pros, and cons. Let’s talk about each of them, their working principles, suitable frameworks, and IDEs.

Java

Java first appeared in 1995 as a language for Internet applications. But today it is used in different development areas like games, navigation, business solutions, and certainly in the mobile industry.

Java is a classical language for Android development. Naturally Java is an interpreted language. This feature makes it cross-platform: to run Java code, a Java VM is needed. In the case of Android, ART or Dalvik* (in KitKat* and older Android versions) plays the VM’s role. But ART uses a quite different approach: it compiles the Java dex bytecode at the app installation phase transforming it into executables for target devices. Here is a detailed explanation of ART’s main features and compilation principles in comparison with Dalvik*.

Google provides a powerful Java infrastructure: Android SDK has an enriched Java API. You can download the SDK tools only or Android Studio—the official Android IDE based on IntelliJ* IDEA including Android SDK tools, platforms, and emulators. It can be installed on Windows*, Mac OS X*, and Linux*. The SDK 6.0 Marshmallow with new features and technologies is already available!

If you prefer to use some of the usual and universal tools you can get the Android Development Tools (ADT) and integrate it into Eclipse* IDE.

Besides SDKs and IDEs, there are a lot of frameworks that help developers, for example, Box2D*, LibGDX*, and Basic4Android*.

C#
C# is known as an object-oriented, Java-alternative language by Microsoft. It is associated with .NET Framework, a Windows* component including Common Language Runtime (CLR) needed to run C# code and class libraries.

In the case of Android, Mono* plays the role of .NET in Windows. Mono is an open source project supported by Xamarin based on the ECMA standards for C# and the CLR.

Mono comprises four main parts: C# compiler, Mono runtime, Base Class Library, and Mono Class Library (classes for Gtk++, Zip* files, LDAP*, OpenGL*, Cairo*, POSIX*, and so on).

Xamarin also proposes a range of instruments for mobile developers: Xamarin Platform for building cross-platform C# apps with native interfaces, Xamarin Test Cloud for measuring the validity on a real device that you don’t have, Xamarin Insights for real-time monitoring on how users interact with your app, and Xamarin University online and offline classes to learn about mobile development. Xamarin framework easily integrates with Microsoft Visual Studio*.

The following frameworks are available for C# programming: Unity*, Monocross*, and Cocos2D-XNA*.

C++

C++ is one of the most powerful languages in the world. There is an endless list of software that can be written on it: Adobe Photoshop*, Illustrator*, Windows OS*, and other Microsoft products like Office*, Visual Studio, and Internet Explorer*, as well as Mozilla Firefox* and some OS X parts.

C/C++ Android development traditionally represents a way to build native apps. Native means the compiler transforms a program code into hardware-specified binaries. So native code must be built for each particular device. It is quite difficult, but these types of apps are known as the fastest. Moreover native apps have direct access to the camera, GPS, or address book and other features of the OS and device.

The described native application development has its own pros and cons. It is important to understand the app you are going to create. Is it a 3D game or simple notebook or an app for calling a cab? If the app needs intensive CPU or GPU computations, C++ coding might be a good idea. But it’s important to keep things simple, instead of making them complicated when they don’t need to be.

If you are 100-percent sure that native featured development can increase performance, you’ll need the Native Development Kit (NDK).

However, completely native Android apps are rare. Usually they have parts written on other languages. For this reason Java provides Java Native Interface (JNI), which calls native functions from Java VM.
Many game engines or other frameworks are aimed to simplify C/C++ Android development. Here are some of them: **Unity**, **Unreal Engine**, **Project Anarchy**, **Marmalade**, **Cocos2D-X**.

Intel provides two tools for optimization and improving performance of Android native apps: **Intel® Graphics Performance Analyzer** (Intel® GPA) and **Intel® VTune™ Amplifier** for Android. Intel GPA consists of three parts: System Analyzer, Platform Analyzer, and Frame Analyzer, which help detect and solve difficulties in your GPU-bound app. [Here](#) are details about each one. In the case of a CPU-bound app, Intel VTune Amplifier will be irreplaceable for finding and fixing hotspots and bottlenecks. Look at the [tutorial](#) to get more information.

**HTML5**

HTML5 is a markup language standard for structuring and presenting web content. Generally HTML5 markup is used with the other key web instruments: JavaScript* programming language and CSS3 (Cascading Style Sheets). The combination allows you to create very complex web applications.

But the World Wide Web is not the only place where HTML5 is used. Nowadays it is widely used in mobile development for creating hybrid applications. So if you are an experienced web programmer and want to work on Android apps, you won’t need to spend time learning a completely new language.

What does “hybrid app” mean? There is a classification that divides all apps into three types: native, web, and hybrid, that is, putting together web features and native UI elements. On the one hand a hybrid app is run by [WebView](#) – an extension of Android View class that enables you to insert web pages in Android apps. It gives the app the ability to connect with the needed hardware capabilities. On the other hand you can’t visually see the difference between a hybrid app and a native one!

Many engines or SDKs are focused on the HTML5 standard: **Marmalade Web**, **Unreal Engine**, **Cocos2D-JS**, **Cordova***, and others.

To cost effectively create hybrid Android apps try **Intel® XDK**, a development environment that provide tools for each development stage—from design to sharing at app stores. It includes an intuitive GUI layout editor called [App Designer](#) and a [Built-in Code Editor](#) that accelerate the development process. Enjoy app testing with [Live Development Tasks](#). For more information about these and other Intel XDK features read the [Getting Started Tutorial](#).

Intel® XDK has a special service for 2D game development. Game Asset Manager helps to easily access, manage, and manipulate game assets. A lot of frameworks support and Apache Cordova plugins and APIs are also included. To get started with Intel XDK game development, see [Getting Started with HTML5 Game Development](#).
Conclusion

This article gave an overview of the most widespread program languages and their place in the Android industry. The main takeaway is that you should define the app concept before finding the appropriate program language. You should ask yourself a question: what the app will be? If it will be a simple calculator you can choose between classical Java, C#, or modern HTML5. But if you’re going to launch a 3D game product with rich graphics, try to realize the native power of C++ or C#.

References

- Android Developers
- Oracle Java
- C++ standard
- C# language specification
- HTML5 - World Wide Web Consortium

About the Author

Anna Belova works as a Software Engineering Intern in the Software & Services Group at Intel Corporation. She has a Bachelor’s degree in Business Informatics and is studying for a Master’s degree in Applied Mathematics and Informatics at the National Research University Higher School of Economics. Anna is interested in mobile and wearable technologies.