Programming Which Way

This topic sheds more light on some basic domains in programming or coding. These domains are ;

* Front End Development
* Back End Development
* Mobile Development

Before we proceed, I would like us understand what programming is and what it entails. Programming is basically the tasks involved in achieving a solution using a computer language. The tasks involved may be analysis, generation of algorithms, implementation of algorithms.

In layman language, Programming is developing a solution to a problem. The solution is what we call a “Program”.

What is a computer program ?

A program is a set of instructions that tells the computer what to do and how to do it. These instructions are what direct the computer on how to solve that problem. And they only way we can communicate with the computer is via languages it understands, which is what we call computer programming language.

Every application that we use everyday is a computer program ranging from your music app, video app, email client , whatsapp, facebook and the likes.

Front End Development

Front-end web development is the practice of converting data to graphical interface for user to view and interact with data through digital interaction using HTML, CSS and JavaScript.

Front-end web development is everything involved in programming the user interface of a web application. Typically it refers to the Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript portion of web site production

It encompasses everything from building a simple page of HTML text to creating complex, responsive HTML5 websites designed to be accessed via various different browsers, devices and screen sizes.

Front end developers are responsible for building out the "client side" in web applications. i.e everything you see when you enter a webpage.

Jobs in Front End Development

According to Indeed.com, the average annual salary for front end web developers is $102K and, at the time of this article, there were over two thousand open, full-time positions available. Practically every organization in existence is online, making the position of front-end developer extremely popular. Find excellent job opportunities around the world for JavaScript Programmers, HTML5 coders and more.

Learn Front End Development

From in-depth HTML5 tutorials to advanced online multi-course programs that combine HTML, CSS and JavaScript, individuals can learn the most up-to-date front-end coding practices. Experts from Intel and the World Wide Web Consortium (W3C), the organization that develops and maintains web standards and guidelines, offer web design and development courses to guide you through building a website. Students can also enroll in the advanced professional certificate program in web development, a multi-course specialization designed to get learners on a path to a career in this in-demand field.

Front-end web development is a great place to start your journey into the world of coding.

Sites

Back End Development

There is a portion of the application the user sees and then—in most cases—the largest part of the application remains unseen. This is the elusive and mystical “back-end.”

In the chapter on web development, we mostly talked about web development as it relates to direct interaction with the end user––what we might call “front-end web development.”

In most significantly applications, there is much more non-user interface code than user-interface code.

Complex systems have all kinds of logic that happen in the background that make them work.

Data needs to be stored and retrieved, business logic and rules need to be followed, and results need to be calculated.

All of this happens behind the scenes.

The back-end developer is the kind of developer who makes all this happen.

Back end developers are responsible for building out the "server side" in web applications.

## What Exactly Is “Back-End Development?”

For the purpose of this chapter, I’m going to consider back-end development to be **any kind of development that doesn’t involve creating code that produces a user interface.**

This could encompass back-end web development, but it might also involve writing APIs, creating libraries, or working with components of systems that don’t have user interfaces or even systems of scientific programming.

In reality, even though front-end development gets most of the glory, **most of the code that exists in the world—arguably the most useful of it—is back-end code** that is never seen by the end user.

Simply put: back-end development is simply writing code that is not seen directly.

## What Do Back-End Developers Do?

What back-end developers do can vary greatly, depending on the size and the scope of the application they are working on.

I’ve held many jobs where I was a back-end developer, working on the business logic in an application, and feeding and retrieving data from the front-end.

In the web development world, most back-end developers concern themselves with building the actual logic behind the application they are working on.

Often, front-end developers will build out a user interface, and back-end developers will write code that makes it all work.

For example, a front-end developer might create a screen in an application with a button to press to get the customer’s data.

A back-end developer might write the code that makes that button work by figuring out what data to fetch from the database for the appropriate customer and delivering it back to the front-end, where it is eventually displayed.

## Primary Technologies and Skills in Back-End Development

While front-end developers need to know a set of tools used to create user interfaces, back-end developers usually have a completely different set of tools and skills that are required to do their jobs effectively.

**One important skill back-end developers need is related to SQL and databases**.

Most back-end systems are connected to some kind of database which stores the data for the application.

It is usually the job of the back-end developer to write, read, and process data from a database or other data source, so having skills like SQL can be extremely important.

Back-end developers—at least for web development—also need to be good with server-side languages for the technology stack they are using.

For example, while a front-end web developer might be focused on HTML, CSS, and JavaScript, a back-end developer might need to know more about PHP web frameworks, Ruby on Rails, ASP.NET MVC, or whatever server-side web development framework is being used to build the application.

Finally, I’d say that a back-end developer needs to know much more about application architecturesince, for the most part, it is the back-end developer who is going to be building out the architecture and internal design of the application.

What skills must I know before becoming a backend developer?

* Understand HTML, Javascript ad Css so as to communicate with the front end guys
* Understand database and hosting evironments on the server
* Understand how the server works
* Good knowledge in at least a programming language

Mobile Development

# What is Mobile Development?

Let’s start out by defining exactly what mobile development is because it’s not as obvious as it seems.

It isn’t just about building phone apps, although that is certainly a large part of it.

**It's doing any kind of development for any kind of mobile device.**

Somewhat of a rhetorical definition, but stay with me here.

What I mean by this statement is that this type of development encompases developing apps for phones, tablets, smart watches, and all other kinds of wearable devices which run some kind of mobile operating system.

It also doesn’t necessarily mean developing purely mobile applications, since even web developers today have to think about how their applications are going to be used and accessed on a mobile device.

## What's Special About Mobile Development?

One of the most intriguing areas of software development has always been mobile development.

Why?

**Because it presents a pretty unique opportunity for a one-person development team to build an actual, usable, meaningful app end-to-end in a relatively short period of time.**

It also represents an entrepreneurial opportunity which is well within most programmers’ reach.

Not to say that an ambitious software developer couldn’t build a web application or desktop application by themselves, but this type of development is so much more accessible because mobile apps are expected to be small and singular in purpose.

Even in the gaming world, mobile applications can be a throwback to earlier, simpler times before 3D graphics and humongous code bases.

Retro-style games from the 8-bit and 16-bit era are somehow acceptable and even endeared on a mobile phone or tablet, where on other platforms they aren’t nearly as wide accepted.

However, app development represents more than just an opportunity for the solo-developer to build their own project—**it’s arguably the future of development**, as mobile devices are becoming larger and larger parts of our lives.

It’s quite possible, at least at the time of writing this book, to become exclusively a mobile application developer, both independently and working for someone else.

In this post, we’ll take a look at what mobile development is, go over some of the major mobile platforms, and talk a bit about the technologies that exist for developing, so that you have a good idea of whether or not it's a good match for you.

## Major Mobile Development Platforms

Throughout computing history, there have actually been quite a few different mobile application development platforms, **but until recently, app development had not taken the limelight and no dominant platforms had existed.**

That all changed with the introduction of the iPhone back in 2007.

I remember back when I first started doing any kind of app development, when the Palm Pilot first came out.

One of my first entrepreneurial ventures—and probably the first application I completely built on my own—was a Magic: The Gathering Life Counter application written in C for Palm OS.

Since then, many mobile experiments have flourished for some time and then died on the vine.

Windows CE seemed so promising, but they just couldn’t get it right.

Blackberry looked like it was going to dominate the world, and perhaps it did for a time.

But today—at least at the time of writing this book—**there are two main contenders**, and then there’s the rest

## iOS

**iOS is quite arguably the “big dog” when it comes to major app development platforms,** partially because it was the platform that finally brought mobile development into the modern day and age by completely transforming the idea of a mobile device and mobile software.

iOS is, of course, developed by Apple, and it runs exclusively on Apple products

At the time of this writing, **iOS runs on iPhones, iPods, iPads, Apple Watches, and Apple TV,** but I expect there will be more devices which will run iOS in the future.

iOS at its core is very Unix-like; it is based on Darwin (BSD) and OS X.

It shares some important frameworks with OS X, and its user interface is based on Apple’s Cocoa UI, which is used in OS X applications, but has been modified and redesigned for touch devices and called Cocoa Touch.

Apple provides iOS developers with several native tools and libraries to develop iOS applications, and, although you don’t need to use Apple’s development tools to build your apps, you do have to have a Mac running OS X to build your application.

**iOS applications are typically built utilizing either Objective-C or the now more popular development language for the platform, Swift.**

## Android

If it’s not iOS, it’s probably Android or both.

Android is the other dominant player in this space.

Android was a little later to the game, first being released in September 2008, almost a year later than iOS, but it has still managed to gain a pretty large share of the mobile market.

Technically, **Android is the mobile OS with the largest, most dominant share of the market,** weighing in at around an 80 percent share compared to iOS’s 18 percent share.

Those numbers are a bit deceiving since Android is a fragmented market, consisting of many different devices made by different manufacturers, running different versions of the Android operating system.

That is the primary difference between iOS and Android.

Android, backed by Google, is open.

iOS, backed by Apple, is not.

Anyone can build an Android device, and it is designed to run on a variety of different hardware platforms and devices with very different form factors and capabilities.

iOS is designed to run, and only runs on, a specific set of Apple devices.

Android is based on the Linux kernel, and the source code for Android is released as open source by Google.

Like Apple, Google provides some native tools for Android development, but again, you aren’t required to use them.

**The native development platform for Android OS application is Java.**

## Everything Else

Everyone else left in the mobile OS market shares a measly, less than 2 percent share of the overall market.

Of the remainders, Windows and Blackberry are probably the largest, but still mostly inconsequential.

**It’s probably a matter of time before both of those mobile platforms completely disappear**.

Because of their small market-share, I won’t even talk about the others since I wouldn’t encourage you to waste your time investing in any platform that has an extremely high chance of dying, but I’ll say that there are some options for developing cross-platform mobile applications—which we’ll talk about a little later on—that will allow you to develop for these marginal competitors at virtually no extra cost.

**I would not ever consider developing exclusively for one of these platforms.**

If you are going to develop a mobile application and become a mobile app developer, choose iOS or Android.

## How Mobile Development Is Done

When iOS and Android first came out, if you wanted to learn how to develop mobile applications for both platforms, you learned how to use the native tools each vendor provided.

For iOS, that was XCode and Objective-C.

For Android, that was an Android SDK plugin to Eclipse, or Netbeans and Java.

Oh my, how things have changed.

Today, there are many more options.

**There are countless frameworks, tools, and entire platforms and ecosystems for developing mobile applications.**

Just about every programming language is supported in one form or another, and mobile applications can even be built to run exclusively in a mobile browser.

Even though there are many options available, we can group them into a few larger categories.

## Native Development

Obviously, we can natively develop mobile applications using the tools the mobile OS vendor provides for us.

As I mentioned before, for iOS, this was initially XCode and Objective-C, but Apple created a new language called Swift, which is now the language of choice for developing iOS apps.

In the Android world, not much has changed except that Google has put out its own Android Development Studio IDE. Java is still the preferred language (although, if you are really brave, C/C++ is officially supported).

When I created my first Android and iOS applications, I built them natively, but I wouldn’t do that again today.

**The biggest issue with native mobile development is that you have to completely rewrite the code for your application for both iOS and Android**. (If you want to support Windows Phone or another small platform, you must do the same for that platform as well.)

It’s not a huge deal, but applications typically need to be supported, so trying to support different versions of an application on two or more totally different platforms can be a bit of a maintenance nightmare.

Also, Android and iOS development is pretty much completely different.

The tools are different, the languages are different, the frameworks are different, and even the development paradigms are different.

If you want to create an iOS and Android version of an application, you have to be prepared to learn two very different, evolving platforms.

Native development does have a few advantages.

The biggest one is raw speed—although some cross-platform frameworks like Xamarin can match it since they compile down to native code. We’ll get to that in a minute.

**Aside from using a framework that compiles down to native code, native code is pretty much going to be faster than any other solution.**

If you develop natively, **you are also going to have better debugging tools**since you won’t be operating with several layers of high abstraction.

You are also more likely to be able to **take advantage of some of the native features of the platform** and get closer to the hardware level. (Although, again, this may not be the case with some of the better cross-platform offerings out there.)

Overall, I think it’s useful to know how to do native mobile development, but I don’t think it’s the best solution for delivering an app to multiple platforms.

## Cross-Platform Frameworks & Tools

The next option is to choose to use a framework or tool that is designed to allow you to build cross-platform mobile applications.

There are many of these solutions to choose from depending on what you are looking for.

Some of these solutions actually produce native code and wrap the real native libraries, so they are just an abstraction over the native language and tools, but they still require you to know and use the native libraries and frameworks.

Other solutions will build a hybrid application that has some native components and some web-based or HTML components and relies on the built-in mobile browser for creating most of the user interface and functionality of the application.

There is an ever-growing list of options to choose from, so choosing can be difficult.

The main considerations to think about when choosing a cross-platform framework solution are:

* What programming language you should use.
* Whether you want to take a native or hybrid approach.
* How many platforms you want your code to support.
* Whether you can reuse code or not.

**Programming Language**

What programming language do you want to use to code the application you are building?

Most cross-platform solutions support a single programming language.

You might not want to incur the learning curve of a new framework, mobile development, and a new programming language all at the same time, so you might want to pick a cross-platform solution which supports a language you already know.

**Native or Hybrid**

There are several cross-platform solutions which compile the final code down to the native format for the mobile operating system and hook directly into the native libraries and APIs.

My favorite one is Xamarin, which allows you to write your app in C# but still get all the benefits and features of a full, natively-built application.

There are many other options to choose from.

Other cross-platform solutions, like Cordova for example, take a hybrid approach where the application is not a native application, but it looks like one.

**Typically, native is going to be faster and look and feel more like the mobile platform the app is running on,** but some of the cross-platform, hybrid solutions are getting so close to native that is can be difficult to tell the difference.

**Platform Support**

Another major consideration is platform support.

Just about all cross-platform solutions support iOS and Android, but some cross-platform solutions also support desktop operating systems like Mac OS X or Windows, and others support the smaller phone OSes or even Raspberry Pi.

If you need to support Blackberry, because that is what your customers are using, you will be constrained to cross-platform solutions which support it.

**However, if you don’t have specific needs for platforms other than iOS and Android, I wouldn’t worry about cross-platform support beyond the two big dogs.**

Games are a different consideration.

If you are doing game development, you might want to pick a tool which supports the widest number of platforms possible.

Tools like Unity 3D allow you to create games that can be run on just about every platform you can think of—even the web.

**Code Reuse**

Finally, you should consider code reuse.

Just because a framework is cross-platform doesn’t mean you’ll be able to write all your code for the application once and have it run on all the supported platforms.

**Typically cross-platform solutions which offer native support will have less code reuse because they will be tied more closely in with the native frameworks**, libraries, and user interface elements and paradigms.

Therefore, you might have to choose between being more native and having the look and feel and design of the mobile OS platform you are running on, or sharing more code.

Recently, though, cross-platforms solutions like Xamarin have come out with solutions which allow you to get some of the benefits of both.

For example, Xamarin uses a common UI library called Xamarin Forms that allows you to have a much larger percentage of code reuse between platforms by creating another abstraction layer on top of the native UI and frameworks of the underlying operating systems.

Ultimately, code reuse depends on what kind of application you are building and how much you want the application to blend in with completely natively-built applications on the platform you are targeting.

## Mobile Web Apps

Finally, you can decide to build a mobile application that is completely web-based.

**Over the years, this option has become more and more feasible as the power and capabilities of mobile browsers have increased and adaptive web technologies have improved.**

With this option, you build a web application, just like any other web application you would build on the web, but you would specifically design it to work on mobile devices.

Many mobile OS browsers even have hooks to support invoking native functionality from inside the web browser, so you can do things like get location data and access things like the camera on the device.

There are even quite a few frameworks out there to help you create mobile web applications which look like native applications when run on a particular mobile OS.

Honestly, **the future is mobile web apps**—we just aren’t there quite yet.

## Mobile Development Considerations

I personally think mobile development is an excellent choice for getting started in software development because it has a **low barrier to entry**and is an area of software development that will **continue to grow over the years**.

Just about anyone can become a mobile developer and even publish their own mobile applications—and perhaps also make some cash.

Plus, mobile applications are by nature limited in scope,**so a team of just one developer can produce a significant application in a relatively short period of time** which can act as a great portfolio of work to help land future jobs or contracts.

I often encourage beginning developers who are having a difficult time breaking into the industry to consider building a few mobile applications and getting them deployed on one of the mobile app stores.

Doing so can help you to have the kind of prerequisite proof that you can actually write code and that you can produce a complete working application.

Also, as I mentioned before, mobile development is likely to only continue to grow over the years, so the future prospects for mobile developers seem really good.

**Jobs in Mobile App Development**

Almost every major company with an internet presence also has an associated app that tailors and extends its services to the mobile user. At the time of this writing, Indeed.com listed over 500 open, full-time positions for mobile app developers in the U.S. with salary estimates ranging from $50K to $115K per year. Top locations in the U.S. include San Francisco, New York and Seattle and some positions offer remote employment opportunities. Learn more about Android and iOS app development and explore job opportunities in the exciting, in-demand field.