Introduction to iOS Development

Recently I was asked by a friend to give them some advice on how to start doing iOS development. This isn't a question I get often, due to my position in the iOS community I am often asked questions about technical aspects of development, but not how to get started with development. I thought this would be a good time to look back on how I got into development and iOS development specifically. This post will focus on the technologies provided and supported by Apple.

Table of Contents

Languages

Objective-C

Objective-C is the most popular language for writing iOS software. All of the frameworks that Apple provides to use on iOS are written in Objective-C (or C). This is language that has the most support for development on Apple's

platforms. Apple provides a lot of good resources and guides and learning the language and development.

- Programming with Objective-C - This is a good place to start if you are not familiar with Objective-C at all and want to learn the basic language concepts and constructs.
- Object-Oriented Programming with Objective-C - Objective-C is a super-set of the C language that adopts a lot of the behaviors of the Smalltalk programming language. This guide details how to follow object-oriented design when working with Objective-C.
- Adopting Modern Objective-C - Objective-C has been around for over 30 years, for the most part many of the language concepts have remained the same but in the last few years there have been a number of language enhancements that have been made by Apple. This guide is for getting up to speed with these changes and how to take advanage of them effectively.
- Concepts in Objective-C Programming - The primary framework that you will be working with on the iOS platform is called “Cocoa”. This guide introduces the common programming concepts used by the Cocoa frameworks (which will be covered in more detail later in this post).

In addition to these resources, there are a number of really good books that have been written on getting started with using Objective-C:

- Objective-C Programming: Big Nerd Ranch Guide
- Learn Objective-C on the Mac
- Learning Cocoa with Objective-C

↑ Parent

Swift

Apple created a new programming language a few years ago, called Swift. This is designed to be a safer and faster language to use for development. Please check out www.swift.org to get the latest information about it.

- The Swift Programming Language ~ iBooks Version ~ (2.2) ePub Version - This is the official language document to learn and understand how to use the Swift programming language. This language is relatively new and is still evolving.
- Using Swift with Cocoa and Objective-C - Introduces how to use Swift with
- Intro to iOS Development in Swift - This is a guide specifically for the starting development on iOS with Swift.
- Swift Programming: Big Nerd Ranch Guide

↑ Parent

↑ Table of Contents

Frameworks

The Objective-C language doesn't provide terribly much to work with on its own. You have access to the C and C++ standard libraries. Swift has a more extensive standard library, but it is still not enough to build user applications with. For the majority of things in iOS development we rely on a set of frameworks provided by Apple.

Cocoa

The collection of Objective-C-based frameworks are referred to as "Cocoa". These provide almost any functionality that you would wish to have in an application. There are over 80 frameworks that are part of this umbrella, that cover everything from low level computing, to audio and video, to basic UI creation and flow. Apple has many how-to guides and API reference pages on their documentation site here.

- Cocoa Programming for Mac OS X
- iOS Programming: Big Nerd Ranch Guide
- iOS Autolayout Demystified
- iOS 9 SDK Development
- Core iOS Developer's Cookbook

↑ Parent

↑ Table of Contents

Tools

Xcode

Xcode is the IDE (Integrated Development Environment) that Apple creates and uses for iOS and OS X development. This is a fairly complex tool, however I have written some extensive documentation for getting up to speed and understanding what you are doing with using it to build applications:

- Managing Xcode
- Using Xcode Targets
- The Xcode Build System

- Guide to xcconfig files
- Xcode Build Setting Reference

Apple has recently updated their guide to Instruments, which is a tool for profiling and finding flaws in software. I highly recommend reading it.

↑ Parent

Reveal

Reveal is an user interface introspection tool. This has been invaluable for me in debugging problems with user interfaces in apps. You can download the trial to this app here.

↑ Parent

Bluemix

Bluemix is the name of the website that hosts the IBM Swift Sandbox. This is a web-based REPL for experimenting with programming in Swift. This behaves similiarly to Xcode Playgrounds.

↑ Parent

↑ Table of Contents

Dependency Management

At some point in a project's life, you will need to start using a tool to manage third party code. For iOS and OS X development there are a couple of approaches to doing this.

Manual Management

The manual management approach to dependencies is completely hands-on. If you are using `git` for version control, you may be using submodules as a means of integrating additional code to your repository. Taking a manual approach to dependency management carries a burden of knowledge of the Xcode build system and integration steps to include it into your project as well as conflict resolution and versioning of the third party code.

↑ Parent

CocoaPods

CocoaPods is the most popular method of dependency integraion on OS X and iOS. In my experience, this has been the best and easiest way for me to integrate dependencies into a project. You can check out a full explanation of what it can do for you here.

↑ Parent

Carthage

Carthage is another dependency manager, that could be described as halfway between manual management and the automated management that CocoaPods provides. It relies on a much more manual approach to integration and configuration.

↑ Parent

Swift Package Manager

The Swift Package Manager (“spm” for short) is a tool specifically for management of Swift libraries. It is currently in beta and will be released with Swift 3, which is scheduled for the end of 2016.

↑ Parent

↑ Table of Contents

If this blog post was helpful to you, please consider donating to keep this blog alive, thank you!

donate to support this blog