iOS Reading Material

Preparing for iOS Programming

Quick Overview

You'll save a lot of time and brain strain on the *Audio App Development and Marketing* module if you get a bit of a head start with learning about and understanding the Apple coding environment.

This means:

- being aware of the constantly changing and updating **programming environment**,
- learning about **Swift**, the language we'll be using,
- reading about how to build simple interfaces and **programs in iOS**.

The best ways of doing this are:

- ▶️ watching some online **tutorial videos**,
- 🖥️ reading some **online tutorials** in case you find their style useful,
- 📘 reading a **book** about Swift,
- 💻 *(optionally)* getting hold of a **Mac**, downloading **Xcode** and writing your first app (for the module itself you will be able to use Macs and iPads in our Mac classroom). Alternatively there are some **online compilers for Swift** that work in any web browser.

Learning how to do ***audio*** on iOS is quite complicated, and is not covered well in the online books and tutorials, so this is what we'll be primarily helping you to do on the module. Later in this guide we'll look at the ***AudioKit*** toolkit, to give you a heads-up on the sorts of sound synthesis and sound processing features we'll have access to.

The standard way of learning the most common (*non*-audio) parts of the language is to work through some existing tutorials. This document guides you to some of the best ones available. So, let's now look at each of the above points in turn, and direct you to the appropriate material to read and view.

N.B. If you can look at ***any*** of the above before the module begins, this will make it easier for you during the module itself. There are many links in this document - too many for everyone to do. So, the most popular ones have been marked with **POPULAR LINK**.

Change change change

Something you should know from the start - **Apple is constantly updating its products** - not just its iPads and iPhones, but the developer tools and operating systems. This makes it very hard to teach! We have to produce teaching materials in advance (and people all over the world are also writing books and tutorials) and then every 6-12 months Apple changes *everything*. Often this is done a few days before our module begins!! So bear with us as things change.

Which versions will we use?

This document is written for the versions that are currently deployed in our MacLab - namely **iOS 18** and **Xcode 16**. Bear in mind that iOS 26 (note the new naming system by *year*) and the next Xcode (also v.26) are now in development (and released to developers in Beta mode over the summer) and will be 'pushed' *during* the time we start teaching the module. However we will stick with the versions above for Semester 1.

If it's changing so often, how can I ever learn anything? Good question!

1) We usually 'lock down' our systems on the *previous* version so that everything is stable for the rest of the year, and it allows us to have working example code ready to go at the start of the course.
2) Luckily, if you read something "from the previous version" it will more than likely still work in the latest version. Much of the best documentation (books, videos etc.) is written for previous versions of iOS, and so the information is nearly all still valid.

If this seems frustrating, please just be aware that this is a fact of life when developing for rapidly changing modern technology, but especially for Apple. Usually not much is *removed* from one version to another, but mostly things are *added*, and almost always you'll need to make a few changes to your code to make it work under the new system. In some years Apple roll out updates to Xcode, iOS or even the Swift language itself multiple times in the year. Think of this as training for real-life industrial development.

A note about the programming language

About a decade ago, Apple introduced a new programming language - *Swift*.
Xcode 16 contains **Swift 6.1** but new updates are likely.

Originally this module used another older language called Objective-C. But around 2018 it seemed that Swift had really taken over as the primary iOS programming language. Objective-C may still be useful for a number of years to come, but Swift is easier, the main online tutorials all use it, and our special audio toolkit (AudioKit) uses it.

Computer Programming

There are two cohorts of students taking this module. Many will have done some computer programming before - and some of you will have experience of several different coding languages. But for others this may all be very new 😲.

a) If you have done some C Programming before then you have a good head start. Apple's Swift language uses many of the principles of C, so you will recognise various features. However, all the *really* useful (and widely-used) features use formats and codewords (and even ways of thinking) that are unfamiliar to C Programmers. So, be prepared to open your mind and think of this as a new language.

b) If you have done some programming before, but *not* C then it is definitely worth treating Swift as a *new* language, and one that it's best to learn from scratch. If you've done any Object-Oriented programming (such as Java, Python or Smalltalk) then you will be familiar with the use of classes, methods and messages, but the actual format - and much of the terminology - may look very new.

c) If you have done little or no computer programming before then you might be feeling a bit scared at this point. Don't worry - because even though you may not have a programming background, you are here because you want to make musical/audio things happen using modern technology. Apple's ecosystem (of programming language, development software, and of iOS devices) makes this not only possible, but enjoyable. Apple state that Swift is a good first language to learn. Be prepared to be bombarded by a lot of unusual concepts, strange phrases, and new code-words. We will show you in the labs how to get a musical instrument working very quickly, and you will learn programming as you go along. The advice is NOT to spend all Summer trying to catch up by learning 'C', but instead it's best to learn Swift as your *first* programming language. If you doubt your own ability, just look at Apple's Everyone Can Code video.

Online Swift Compilers

You can 'follow along' with the code examples using this Online Swift compiler**,** or **this one** **or this one, which work in any browser (i.e. on PCs too)**. Just to be clear, these won't work for our specific *Audio* code examples or for creating visual-based apps on iPhone/iPad (you'll need a Mac for that in order to run Xcode) but they're absolutely fine for learning a bit of Swift code.

This is what the second one looks like:

Learning about Swift - Videos, Websites, and Books

Here is a set of resources that you can browse to help you learn Swift. Remember- you don't need to watch or read *all* of these. But it IS worth flicking through them to see whether you like the style of teaching, then pick one to work through in more detail.

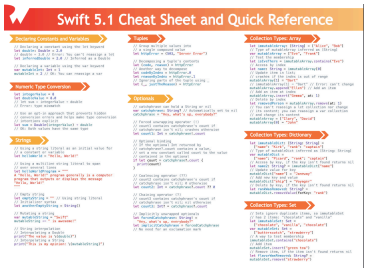
Thoughts about learning a new language

Please note - you do NOT *need* to complete an entire tutorial, course or book on Swift before you start our course. As an analogy, imagine you were visiting a new country whose language you did not know. You can get a phrasebook, learn some common words, then just *have a go* - and people will be pleased to help you. You do not need to become an expert before you travel. And so it is with programming - if you can learn a bit about the language before the module begins, this will help you to understand the lectures better. Please do not feel intimidated or disheartened if you read part-way through one of these guides, then find yourself getting bogged down in detail and concepts. Just reading a bit will help!

Recommended way to start learning Swift by Video

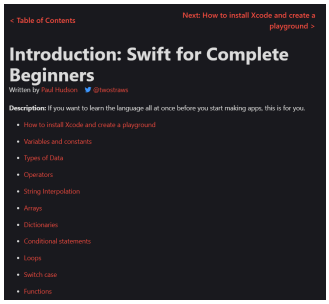
Coding with Chris does some great tutorials, including **TOP LINK** one big video tutorial (3hrs) introducing you to Swift. This is accessible to most students, takes you through the concepts of programming, and can be done on any machine using an online compiler.

Reading about Swift



Most people benefit from taking some time to *read* about Swift, and to try out some simple concepts for themselves.

For those familiar with programming, you might find this **one page Swift summary** by **Ray Wenderlich** to be helpful or this extended cheat sheet.

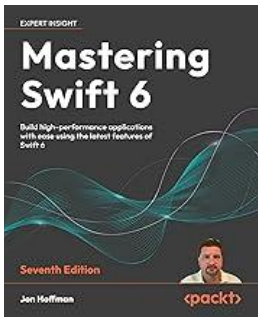


Author **Paul Hudson** has written this **POPULAR LINK** Swift Tutorial which again can be run on an online compiler (despite it beginning with a guide to installing Xcode).
It's a guided tutorial - like the Coding with Chris one mentioned above - but in readable form rather than a big video.

Official Apple Documentation

Apple's website holds the authoritative introduction to the language. This is more suitable for people who can already code to learn the Swift terminology, and it's really *not* beginner friendly. Browse the **POPULAR LINK** 'quick tour' first, or look at the indexed 'Language Guide' which takes you through all the elements of the language.

Read a book



You could buy a book on Swift. For example:

> Mastering Swift 6: Build high-performance applications with ease using the latest features of Swift 6, 7th Edition

This book assumes you have a Mac (running Xcode) but actually most of its code examples can be run in the online Swift compilers mentioned above.

Programming resources for making iOS apps

The resources above are all about Swift as a general-purpose programming language in its own right. But our course is about how to use it to write interactive *iOS* apps that can run on iPhone or iPad. This section shows three methods that keep coming top in many people's recommendations that it's worth recommending them.

A word about these external resources

Please note that in these books and tutorials there is very little about Audio - *we will* teach you about that in the module.

This is just to get you familiar with the words and the concepts involved with app development. You **don't need to become an expert** over the Summer, and on the module I will revise the basics for everyone. (You can just feel that nice glow of knowing that you understand some of it already, rather than everything being new and scary).

First though, let's look at the different ways that you can put things on your iOS device screen, because this is an important factor when choosing a book or tutorial series.

A note about Graphics in iOS

It may come as a surprise that there are many (very different) ways to get graphics on the screen of your iOS device when programming. These are the most popular, but most people have to make a choice between *Storyboards* or *SwiftUI*:

Storyboards - This is the standard way that people have built apps for the last 18 years. It involves dragging items onto a mock-up screen (e.g. placing buttons and sliders) and then linking these to code that you write. This is technically called **UIKit** (**U**ser **I**nterface **Kit**) and is a very popular way of working. Beginners tend to like it because you can physically move things around and it feels very practical.

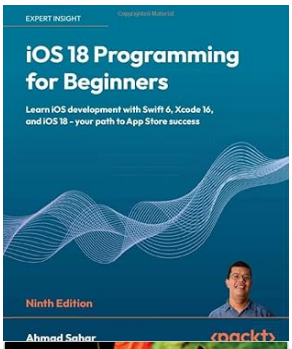
SwiftUI - Then in 2019 Apple introduced a new way of making the interface for your app. It's called SwiftUI (**Swift** language **U**ser **I**nterface), and it lets you create graphical user interfaces by *coding* them. This way of working is increasingly popular, and people who have done a lot of coding before tend to love it.

SpriteKit - This lets you make dynamic and interactive content and is great for games or novel interfaces. We do a little bit on the course, and give you plenty of links for learning it in more detail. I'd recommend NOT learning this over the summer as it's much more optional.

1) Chris' Video Course

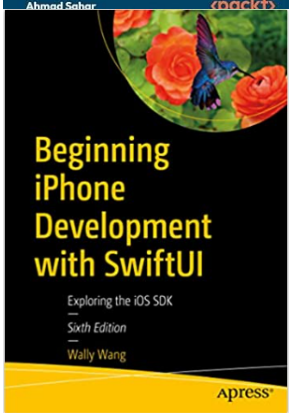
Coding with Chris has also produced [this 30 minute video using SwiftUI] to make a simple app for absolute beginners. Worth a look perhaps? (More info is available on his [website]).

2) iOS books to buy/borrow:



[iOS 18 Programming for Beginners]

This book is an expanded version of the Swift book shown earlier. It's up to date with the latest versions of Swift, retails for about £34 (or £27 for the Kindle version). For its interface design it uses Storyboards, but introduces SwiftUI later on.



[Beginning iPhone Development with SwiftUI: Exploring the iOS SDK]

This book is part of a series that has been constantly updated as Apple technology has developed, which is why it's more expensive[1]. It's a super-thick book, really well illustrated, and many people in the past have said it is the definitive way of learning about iOS programming. The PDF version is available from the publisher Apress (quite cheaply if you have already bought the book) and is full colour and you can have it on your screen while you're working. This version of the book has now been updated to do all the graphics in the programmatic **SwiftUI**. In our module, we give you the *choice* of whether to design your interface with Storyboards OR SwiftUI (or SpriteKit).

3) The Stanford University iOS programming course

Paul Hegarty has become world-famous for his excellent lectures and live demos about iOS coding (and the principles behind it). It's called "Developing iOS 11 Apps with Swift". This is a few years old now and you can **watch the video series on YouTube** and find the lecture slides and assignment sheets **here**. This version of the course uses Storyboards. What really makes this good is watching a world expert explain the whole coding process and language.

Warning: Do not feel intimidated! This Stanford course is ***not*** meant for beginners, and there is an expectation that if you don't know much about object-oriented programming, you'll find it easy to get lost. However, why not just give it a go and see how you get on?

NEW: SwiftUI version. Stanford has released the 2023 version of this course - this time using the new SwiftUI method for coding graphics - as an online set of lectures. [This is the link to the lectures and slides]. They may consider uploading the 2025 course, but it's not there yet.

---

[1] As I write this Amazon are running an offer on this at £31, making it cheaper than the above book

AudioKit: How to do Audio programming on iOS

We have teamed up with the developers of AudioKit, to provide you with some unique learning resources.

We have developed a set of tutorial Audio examples, which you can download in during Semester 1 when the module runs. They will show you step by step how to use AudioKit and Swift to:

- Make an audio oscillator and control it
- Add user-interface elements (buttons, sliders etc.) for real-time control
- Play sound files
- Mix multiple sound sources together
- Trigger and loop multiple sound files, thus creating interactive instruments
- Add audio effects such as reverb, chorus, delay and filtering
- Create synthesis elements and complete synthesisers.

A note about AudioKit versions

AudioKit has to grow and change as the iOS environment and Swift language develop. This year our course will use **AudioKit v5**, which works with **Xcode 16 (and 15)**. I know this is a lot of versions to remember, but you sometimes need to match up the correct versions for them to work together properly 😕.

PLEASE NOTE: You do *not* need to specifically download AudioKit, because we have *included* it within the Lab Examples that we have prepared for you. But later in the module, we get you to set up everything from scratch.

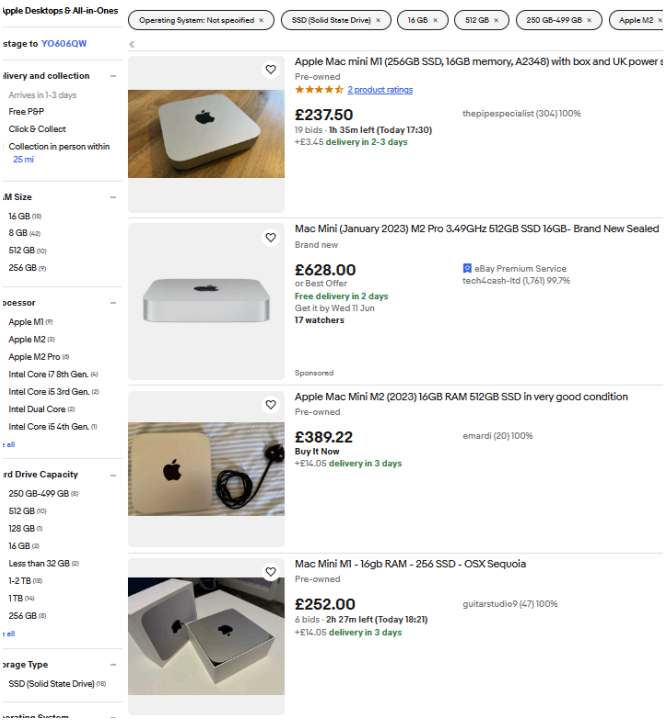
If you're interested in seeing what AudioKit can do, you can take a look at their AudioKit documentation site. Again, if you're new to all this, don't be intimidated as we'll be guiding you through this. But you might be keen to at least see some of the components available.

AudioKit also maintain an industry-facing Pro Site, which is less about the coding, and more about what apps have been built with it.

What Machine should I use?

In our Mac Lab we provide a number of high-spec machines which are suited to developing Apps. **You can work here (and get full out of hours access) to complete your iOS work.**

In my experience though, a few people seem to want to get hold of their own machine to work in their own space and time. Some people already have Macs. Others decide that this is the opportunity they've been looking for to get one or ask nicely for a present. Still others have borrowed a family Mac, or in a couple of cases "a spare one from a friend", so choose your friends wisely!



Xcode 16 will run on most machines from about 2020 onwards (as it requires a modern version of the Apple operating system and an **Apple chip** (M1, M2 etc)).

You need a reasonable amount of memory. Try to get a Mac with at least **16GB RAM** and with an **SSD** drive (ideally **512GB**, but definitely 256GB or more), as this makes Xcode run so much faster. (On an old-style Hard drive or with less than 8GB RAM you have to wait ages for everything!!).

One of the best value ways of getting your own Mac is to get a Mac mini from somewhere like eBay. This is a screenshot from eBay showing recently sold Mac mini's from 2020 with at least that minimum recommended specification.

Clearly if you have a Mac mini you'll *also* need a keyboard, mouse and screen.

Like most Apple technology, things keep moving on and changing, and the operating system version is no exception. For development, make sure your Mac is running the latest operating system. Make sure you get the latest release version of Xcode (so avoid 'Beta' versions for now). At the time of writing this is Xcode 16 running on **macOS Sequoia 15.2**).

Windows alternatives

Some students have had success getting the **Mac development environment running on a PC.** Be warned, it's not straightforward, will take some time and experimentation, and it is not guaranteed to work. If you're up to trying this please see this super article by Coding with Chris all about exploring the different ways to get Xcode running on a Windows PC.

Also please remember that in the labs we will only want to help you with your apps and coding; we are *not* offering help to get Xcode running on Windows - that's a project for you only if you would like to try it and feel technically confident to give it a go.

A quick heads-up on AI

I will be using AI at various points throughout the course to show you how you can use tools such as ChatGPT to:

- Help you de-bug code
- Teach you various bits of code
- Come up with marketing suggestions for your app

I'll also be very clear about what I *don't* want you to do. If you let AI do *too much*, you're not learning, and you'll ultimately put yourself out of a job. Equally if you *don't* use it at all, you'll miss out on an amazing tool that is being used all over the world. You're going to learn how to code with AI as a partner, making you future-proof !

In addition, Apple has just announced widespread integration of AI into its iPhones and iPads and also with code generation as an upcoming part of the development process in Xcode.

In Summary

Work through these links, and see for yourself which materials are best for you. Set yourself a programme of learning, particularly what I outline at the very start of this document. Try doing a bit each day. Or maybe dedicate one week to exploring Swift.

If for whatever reason you don't get a chance to work at this over the Summer, don't worry - but you'll have to set aside a reasonable amount of time for reading and trying things out in Semester 1.

See you in the first lecture,

Andy Hunt 9th June 2025

Glossary

- **App** - short for ‘application’ - the pieces of code that you write and that run on mobile devices
- **iOS** - the operating system for Apple devices (there’s now a related one called iPadOS especially to make use of the bigger screen)
- **Swift** - the programming language we use when coding on iOS
- **Xcode** - the program we use to do all our development (app programming). It’s an example of an **IDE** (Integrated Development Environment) and so you'll need this running on a Mac computer of some sort.
- **Storyboards** - the traditional graphical drag-and-drop method of coding the user interfaces in an iOS app
- **SwiftUI** - a relatively new code-based way to design user interfaces. For your app, you’ll be asked to choose between SwiftUI and Storyboards.
- **SpriteKit** - a toolkit for making games and dynamically responsive interfaces
- **AudioKit** - the library of audio commands that we use to make sound.