

**S2021 MAD400-75 iOS Development**

**Professor: Manhar Kapoor**

**Student Name : Sarfaraz Ahmed**

**Student ID : A00107385**

**Asssignment 1- Part2:** Write a technical document regarding Xcode, Guide to Xcode or introduction to Xcode. By reading your guide, people will learn the xcode. Install or upgrade Xcode as needed. Explain Xcode and its capabilities. Use and explore the xcode interface. Set up fronts and colors, use the tab bar, options, editor, project view, developer documentation, Quick help inspector, and other tools.

**Xcode :**

Xcode is the Mac app that you use to build apps for iOS, macOS, tvOS and watchOS. You use Swift programming, and the many tools inside Xcode, to build applications for iPhone, iPad, Mac, Apple TV, and more.

Xcode is an IDE, an Integrated Development Environment, which essentially means that Xcode includes many additional tools for development. A few examples are: a debugger, source control, device management, iPhone Simulator, profiling tools, Interface Builder, documentation, and much more.

**System Requirement :**

Minimum system requirements for Xcode

* A Mac with macOS 11 (Big Sur)
* 4 GB RAM, but 8+ GB is more comfy
* At least 8 GB of free storage space†
* A 2013-2015 or newer Mac, MacBook, iMac or Mac mini

Xcode only runs on macOS, which means you need a Mac if you want to build iOS apps with Xcode. Xcode won’t run on iPad. A few [alternatives for Windows/PC](https://learnappmaking.com/xcode-for-windows-how-to/) are available, but they’re far from ideal.

## Downloading, Installing and updating Xcode :

It’s easiest to download and install Xcode via the Mac App Store. Here’s how:

1. Search for Xcode in the Mac App Store, or use [this link directly](https://apps.apple.com/us/app/xcode/id497799835?mt=12)
2. Click Install(or Get)
3. Wait for Xcode to complete installing, which may take a while!
4. Open Xcode via your ~/Applications folder or via Launchpad

You can also install older versions of Xcode. This is especially helpful if your Mac isn’t supported by Xcode. Here’s how you do that:

* [Get a free Developer Account](https://learnappmaking.com/how-to-create-a-free-apple-developer-account/) with your Apple ID
* Go to [developer.apple.com/download](https://developer.apple.com/download/) and sign in
* Click Release or More at the top-right of the page
* Use the search field to find previous releases of Xcode
* When you’ve found what you’re looking for, download the .dmg to begin the installation

The latest version of Xcode is 11. If you want to check what version of Xcode you currently have, just launch it and in the welcome screen, you’ll see it clearly stated.

Xcode 11 brings lots of new features such as support for Swift 5.1, Apple’s latest iteration of the Swift programming language, SDKs for iOS 13, tvOS 13, watchOS 6 and macOS Catalina 10.15.

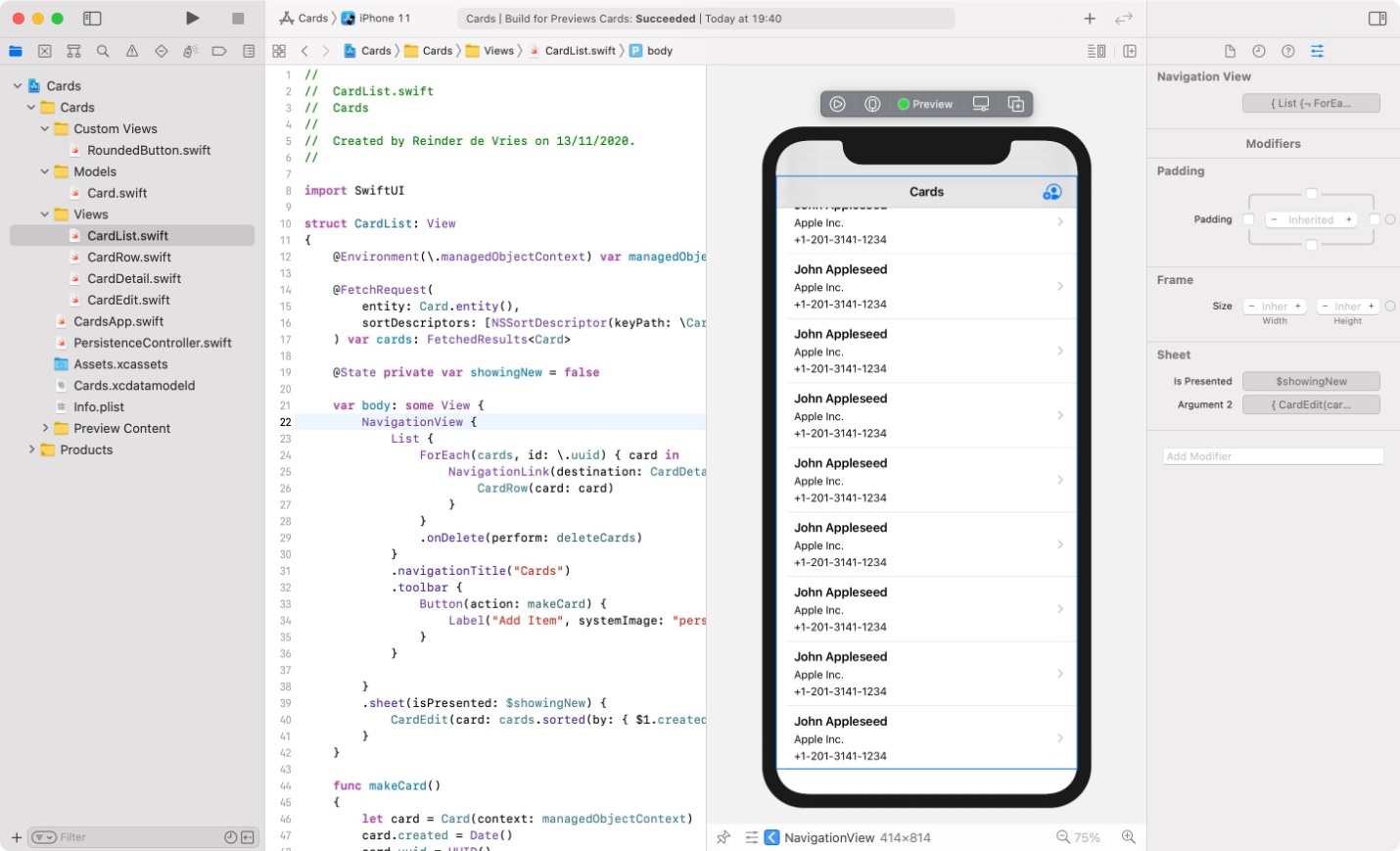
Each of those platforms has changed since their previous iteration and Xcode 11 allows you to build apps to utilize these new features. In terms of how Xcode itself has changed as an IDE (integrated development environment), there are many new features to make it easier for you to build software.

In terms of how Xcode itself has changed as an IDE (integrated development environment), there are many new features to make it easier for you to build software.

## Xcode and SwiftUI

Xcode is upgraded once a year, around Sept-Oct, at the same time the new major version of iOS is released. Every update of Xcode brings improvements, [new features](https://learnappmaking.com/xcode-minimap/), bug fixes, and access to the latest SDKs. Throughout the year, a number of smaller updates to Xcode are made, including [updates for Swift](https://learnappmaking.com/how-to-keep-up-with-swift-changes/).

A notable newcomer in Xcode is [SwiftUI](https://learnappmaking.com/swiftui-getting-started-how-to-ios-swift/). With SwiftUI, you can declare the User Interfaces (UIs) of your iOS apps, and their behavior. It’s the new way to build UIs – and it changes how we think about building User Interfaces for iOS apps.



An exciting aspect of SwiftUI is that you can use Swift programming to build User Interfaces.

quick example:

struct ContentView: View {

var body: some View {

Text("Hello worl!")

}

The above code pronounces a view, which will put that notable Hello worl! text on screen. As yet, you'd make a Storyboard-based view regulator to show anything on screen in your iOS application. You can fabricate those view regulators with Interface Builder.

Coding a UI with code is novel and invigorating, but on the other hand it's scheduled to be more useful. What's more, you would now be able to submit UI code with Git — without consolidate clashes — and that was beyond the realm of imagination previously.

SwiftUI is new, and despite the fact that it's openly delivered as creation programming, it'll need to go through a selection stage prior to turning into the new defacto standard to fabricate UIs. All things considered, SwiftUI will see a selection bend like Swift in 2014, for example moderate yet consistent development.

Until SwiftUI takes over Storyboard-based UIs, figure out how to work with see regulators and storyboards, just as with SwiftUI. In addition, SwiftUI coordinates well with existing UIKit-based perspectives and libraries, which implies that UIKit-based perspectives stay pertinent for significant time. It's brilliant to learn the two methodologies.

**Using Xcode :**

Let’s take a look at Xcode! First, make sure you’ve started Xcode on your Mac. You’re greeted with Xcode’s Welcome Screen:

[](https://learnappmaking.com/wp-content/uploads/2020/11/xcode-tutorial-1.jpg)

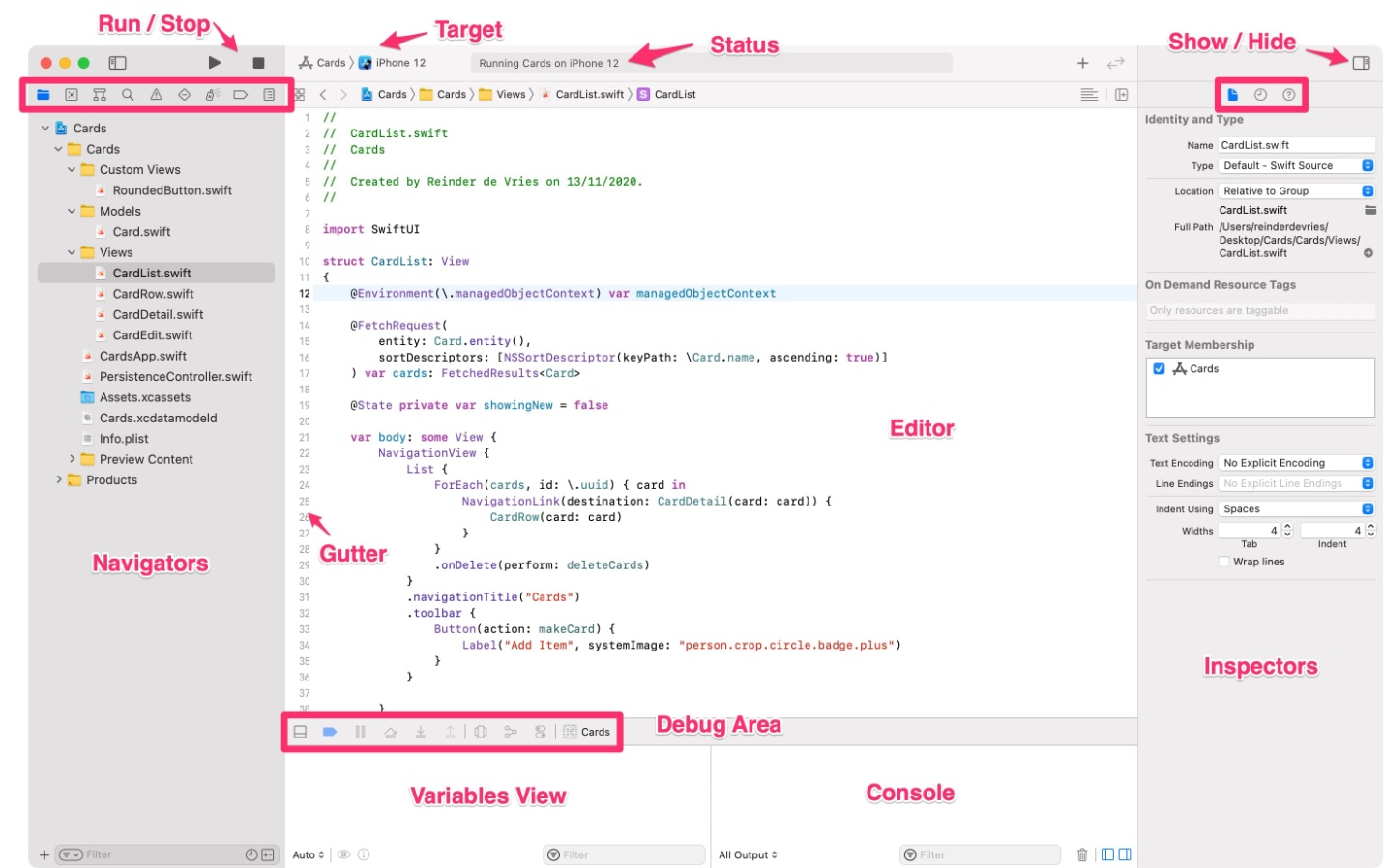
Source : google imagaes

Here’s what you can choose to do:

* [Start a new playground](https://learnappmaking.com/xcode-playground-get-started-with-swift/) to code Swift
* Create a new Xcode project, like [building an iOS app](https://learnappmaking.com/how-to-make-an-app/)
* Clone an existing project from a Git repository
* Open any of your latest projects, on the right

You can also open any other project with the button in the bottom-right, or choose to hide the Welcome Screen next time Xcode starts. If you’ve got many Xcode projects, just opening them via Finder might be more productive.

Next, when you open an iOS app project in Xcode, you see something like this:

[](https://learnappmaking.com/wp-content/uploads/2020/11/xcode-tutorial-2.jpg)

Source : google images

Let’s go over the Xcode UI one by one. First, we’ve got 4 major areas in Xcode:

1. On the left, you see the Navigators. They help you locate resources in your Xcode project, such as Swift files, issues, breakpoints, and more.
2. In the middle, you see the Editor. This is where the magic happens! You write Swift code here. (In Interface Builder, the middle part is where you build UIs. More on IB, below.)
3. On the right, you see the Inspectors. They help you inspect and adjust attributes of files, UI elements, etcetera. Inspectors are the most useful in Interface Builder.
4. At the bottom, you see the Debug Area. You use this area to debug your app; to see debug output, and what’s going on if your app crashes or has a bug. Debugging is an important aspect of app development.

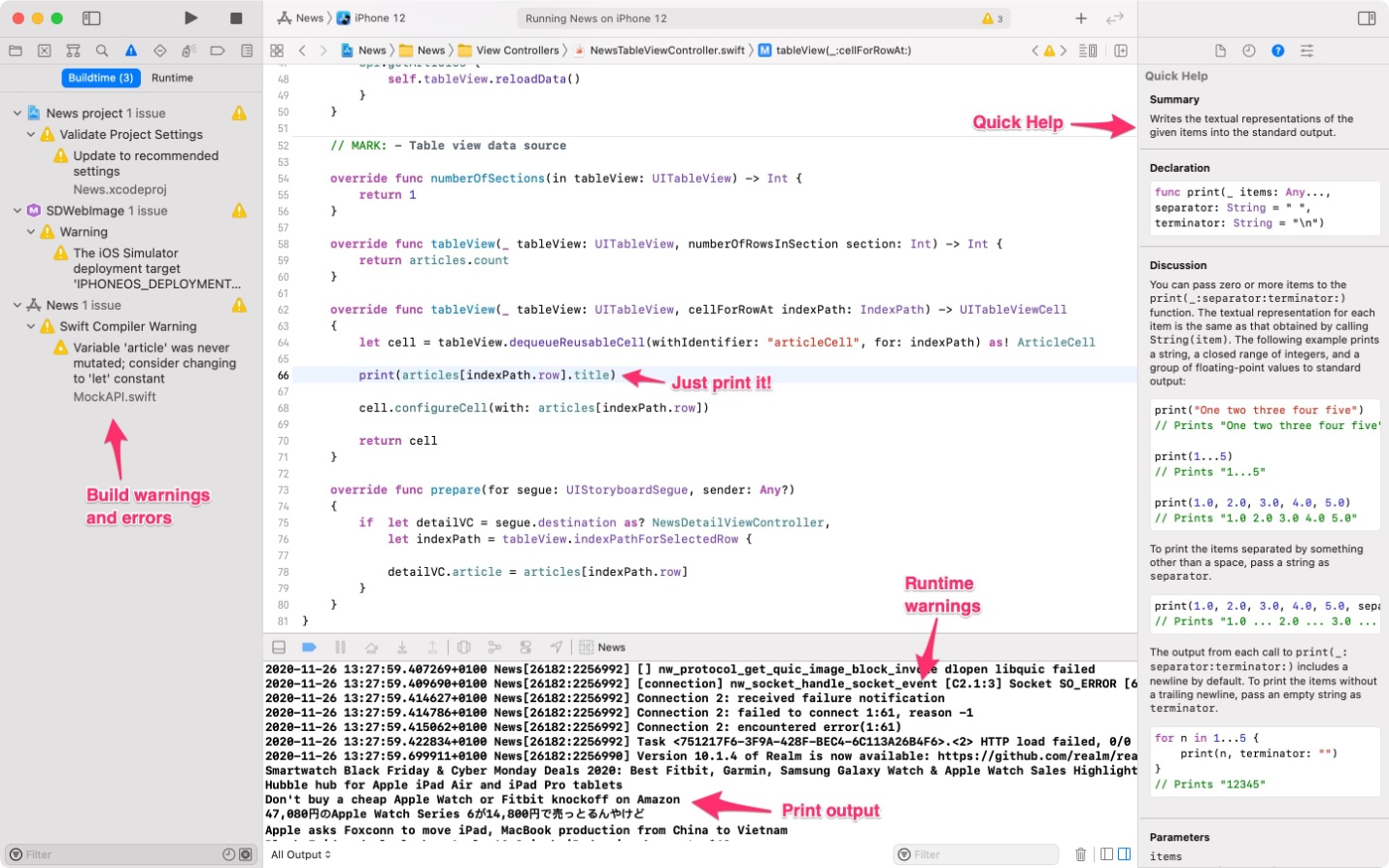
Xcode’s UI also has a few other useful bits and bobs, such as:

* At the top right you see a Play and Stop button, and a dropdown item next to it. With these buttons you can build your app, after which it starts running on the given target. In other words, first select “iPhone 11 Pro”, then hit Play, and your app starts in iPhone Simulator. Neat!
* At the top, in the middle, you see a status bar. With it, Xcode will tell you what it’s doing. It’ll also inform you about errors in your project.
* At the top-right, you’ll see a few buttons that let you show or hide parts of Xcode. If you want to focus on some Swift code, you can hide the left and right panes, for example. The + button is the Library, which includes Swift code snippets (and more, in Interface Builder). The button with the two arrows enables Xcode’s Code Review mode, which lets you compare Swift files side-by-side.
* The code editor in Xcode has two neat functions: a breadcrumb, and a minimap. The minimap is a global outline of your code, and it works like a scrollbar. The breadcrumb, or jump bar, helps you navigate the hierarchy of your project. You can also use it to jump to a particular function or class. And the < and > buttons on the left let you go back or forward to a previous file, like in a web browser.

## Xcode’s Debugging Tools :

When you’re coding an app, you can expect to spend a fair chunk of time on [finding and fixing bugs](https://learnappmaking.com/debugging-with-xcode-ios-how-to/). What tools does Xcode have to make debugging easier?

1. A Console, to show debug output as your app runs
2. A Variables View, a way to inspect the values of variables at runtime
3. Breakpoints, an approach to halt execution of your app at certain points, to inspect what’s going on
4. Instruments to inspect memory usage, CPU utilization, memory leaks, and much more
5. A View Hierarchy debugger, to check out views’ dimensions and positions at runtime

Let’s look at a few of these tools one by one.[](https://learnappmaking.com/wp-content/uploads/2020/11/xcode-tutorial-16.jpg)

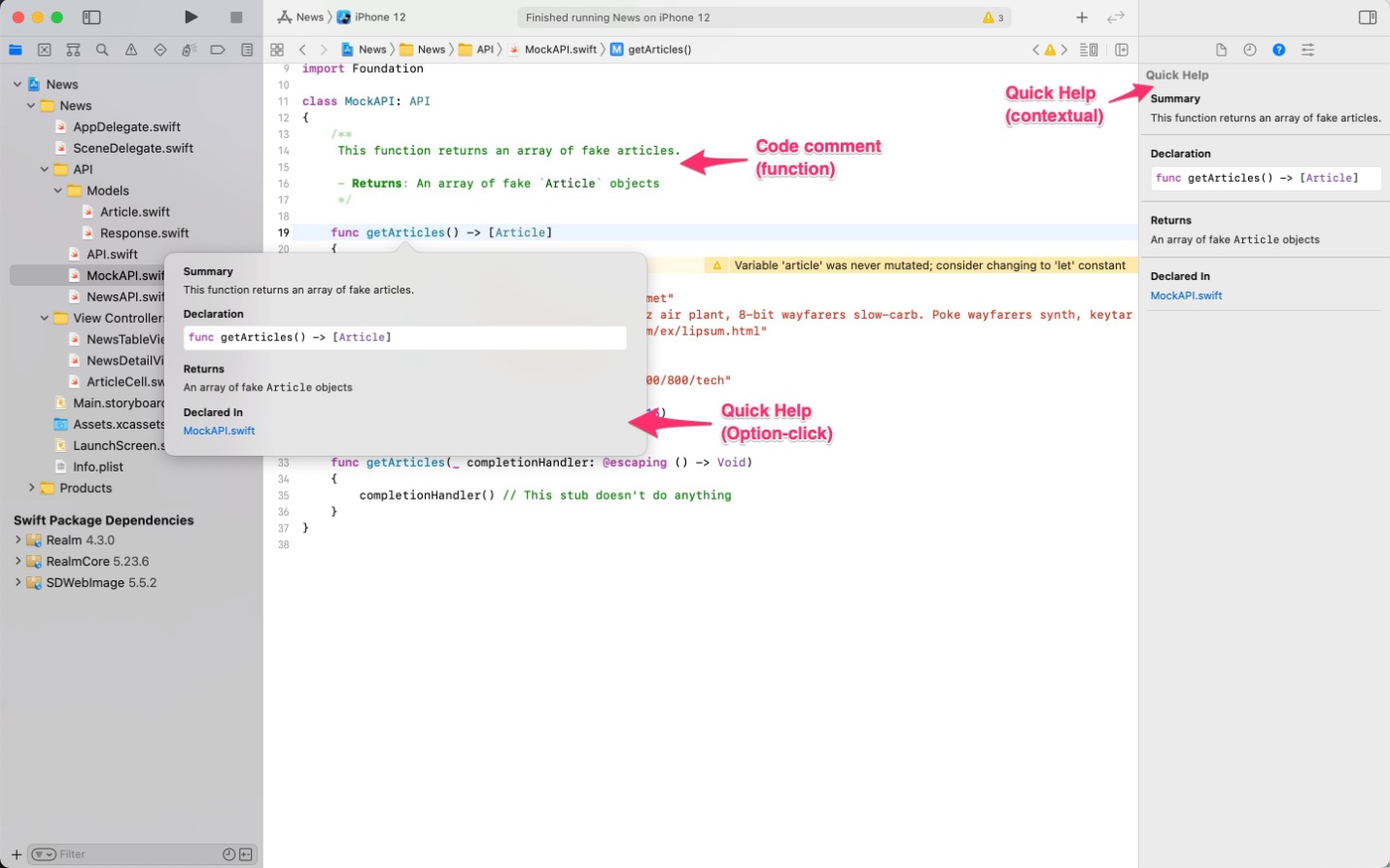
* We’ve used Swift’s [print() function](https://learnappmaking.com/print-swift-how-to/) to print some output: article titles
* This output is shown in the Console, below the editor, as the app runs
* We’re also seeing some HTTP error in the Console, which is useful info

## Xcode’s Quick Help, Documentation and API Docs :

Xcode includes documentation, and documentation tools, right within the IDE. We’ve already seen the Quick Help Inspector in this tutorial, but there’s more!

1. Documentation about your code, such as quick help annotations and code comments
2. Documentation about Apple’s code, such as the Developer Documentation and API docs

Xcode recognizes this function comment – often called a doc block – and understands that I’ve documented what the function does, and what it returns. When you hold the Option key and click on a function, it’ll show a popout window with the available documentation for that function.

[](https://learnappmaking.com/wp-content/uploads/2020/11/xcode-tutorial-18.jpg)

You can use Option-click with any kind of symbol, i.e. functions, types, variables, classes, properties, protocols, etcetera. And it also works for Apple-provided SDKs, APIs and code! Like this:

What if you need in-depth information about a technique, approach or framework? Apple’s Developer Documentation is built into Xcode, so you’ve always got that right at your fingertips.

You can access the Developer Documentation in Xcode like this:

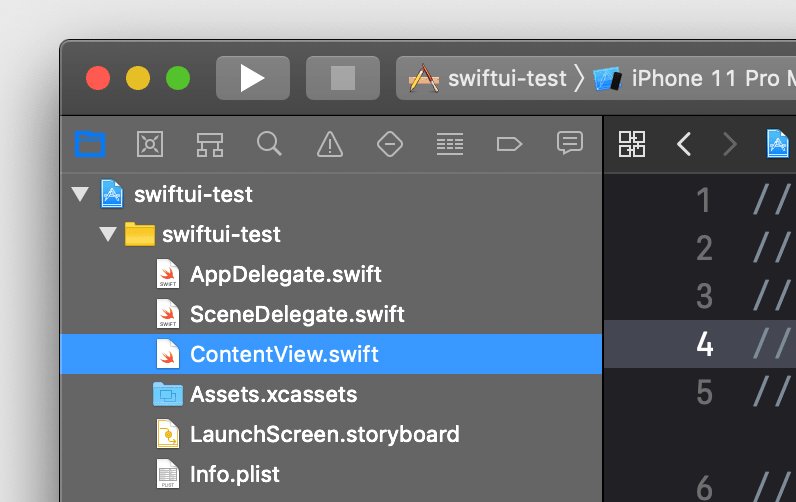
* Press Command + Shift + 0 (that’s a zero)
* Via Window → Developer Documentation
* Via occasional buttons that say Search Documentation

When the docs have opened, you can use the search bar at the top to find what you’re looking for. You can also use the list on the left, which includes tutorial-style guides about various topics. This is quite useful if you’re looking for a general direction, and not for a specific class or function.

Xcode’s documentation can be made available offline, via Xcode’s preferences. You can also find the Developer Documentation online, via [developer.apple.com/documentation](https://developer.apple.com/documentation/). A great alternative to Xcode’s built-in documentation is [Dash](https://kapeli.com/dash), which organizes your developer docs in one useful tool, and it’s offline*.*

**The Xcode Project Navigator**

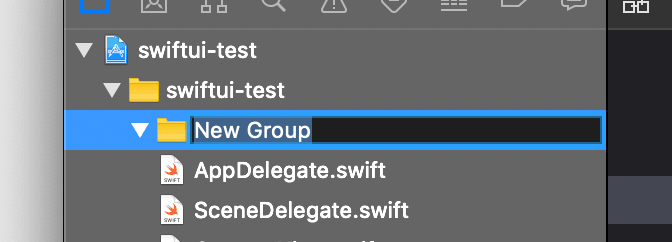
The Project Navigator is where you’ll see all the files associated with your project. This is the default tab you’ll see when you create a brand new Xcode project.



Xcode Project Navigator

Organizing your files

Within the Project Navigator, you can also create Groups to organize your files in. Think of them like folders. You can create a new Group and then drag your files into them to organize your project a little bit.



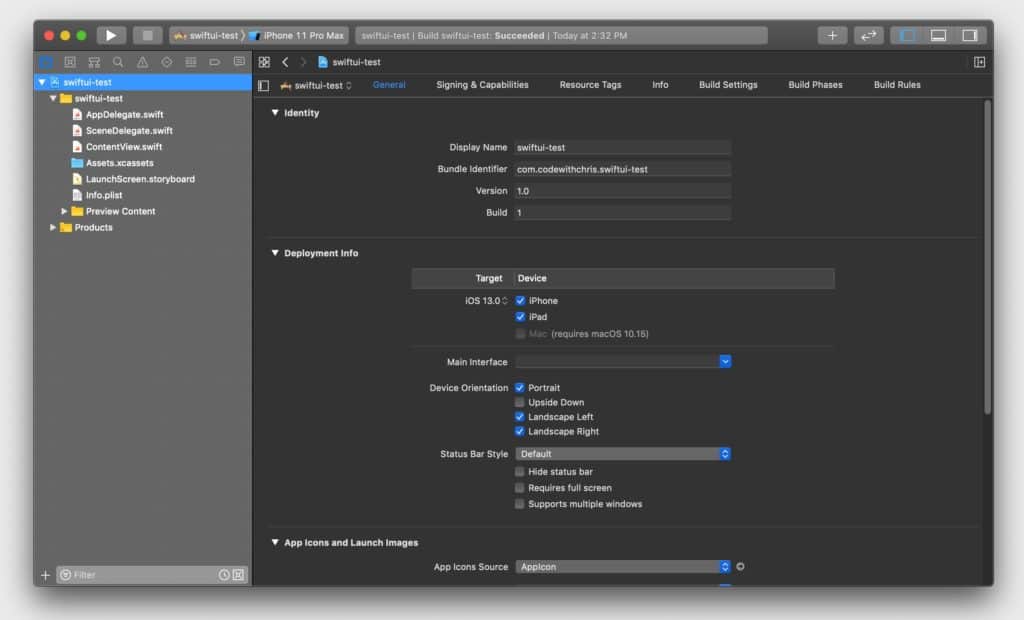
Creating new groups to organize your Xcode project

Creating and adding new files

You can also right click on the Project Navigator and create new files or add existing files to your project. Or you can drag folders or files from your computer directly onto the Project Navigator and will popup a dialog asking how you want to add the files.

**Xcode project properties**

The root node of the project navigator is your Xcode project file (indicated by the blue icon). If you click that, the project properties will open in the editor area.



Xcode project configuration and properties

You’ll visit the project properties screen quite a bit during the process of building your app. In this screen, you can configure things like:

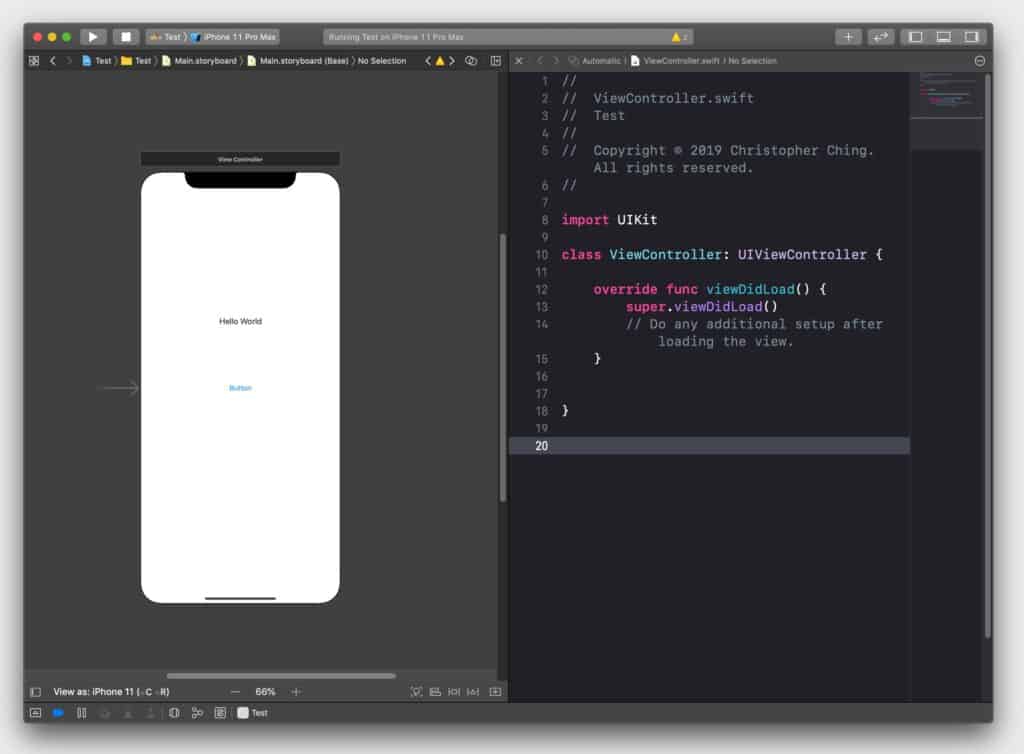
* Set the name of your app
* The bundle identifier (your unique ID for the app)
* Allow your app to work on multiple device orientations
* Set the version of your app
* Set the minimum required iOS version that your app can be install on
* Add additional Apple frameworks and libraries
* Indicate your code signing options which are mandatory for submitting your app to the App Store (if you have code signing errors, this is where you’d fix it!)

**Assistant Editor**

The Assistant Editor view which will show you the accompanying file to the file you’re currently looking at. It looks like a two pane view.

If you’re working with Storyboards, the Assistant Editor will be helpful for connecting user interface elements from the Storyboard view to the code file so that you can manipulate those elements via code.

For example, if you’re looking at a view in the Storyboard, the right pane will show you the linked class file for that view. Each pane also has independent jump bars, so you can also use that to change what file each pane is displaying.



The Assistant Editor is useful for connecting elements in your storyboard to your code.

## Other tools in Xcode

The Xcode app itself can be extended with plugins, called Xcode Source Extensions, and many developers have taken the opportunity to add useful features to Xcode. There’s a whole ecosystem around building iOS apps, with plenty of 3rd-party tools.

* [Fork](https://git-fork.com/), a free and open source Git client/GUI
* [Fastlane](https://fastlane.tools/), an entire suite of app automation tools
* [Balsamiq](https://balsamiq.com/) for wireframing/mockups and [Sketch](https://www.sketch.com/) for vector graphics and UI design
* [Reveal](https://revealapp.com/), a view debugging tool for iOS
* [SimPholders](https://simpholders.com/), productivity tool and launcher for iPhone Simulator
* [xScope](https://xscopeapp.com/), a visual inspector tool for apps
* [Bugsnag](https://www.bugsnag.com/), an error reporting and QA tool for production apps