Chapter 1

Introduction to iOS Development

Objectives:

- Touch on the history of iOS and the devices that support this operating system.
- Understand the different types of Apple Developer accounts.
- Introduce the iOS SDK.
- Discover the basic programming paradigm for iOS apps.
Chapter Overview

iOS is both an operating system and a Software Development Kit (SDK) to write applications for the device. In this chapter, you are introduced to the basic concepts and tools of the iOS SDK. Consider this to be the “50,000-feet view” of the entire development environment. All of the concepts introduced in this chapter are explored in greater detail through the rest of the course.
Introducing iOS

- iOS, originally called iPhone OS, is an operating system for Apple’s “limited devices.”
  - iPhone OS was released with the first iPhone on June 29, 2007.
  - The current major version is iOS 7, with an initial release on September 18, 2013.
- The operating system runs on several Apple devices, including the iPhone, iPad, and iPod touch.
  - Unlike Windows, which may run on third-party hardware, iOS is strictly limited to run on Apple's devices.
  - Not all devices are able to support all features provided by the current version of iOS.
  - Some devices have been decommissioned and no longer receive iOS updates; others are current, but may lack the hardware to support specific features.
• **iOS 7 Supported Devices:**

<table>
<thead>
<tr>
<th>Device</th>
<th>Initial Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone 4</td>
<td>June 24, 2010</td>
</tr>
<tr>
<td>iPhone 4S</td>
<td>October 14, 2011</td>
</tr>
<tr>
<td>iPhone 5</td>
<td>September 21, 2012</td>
</tr>
<tr>
<td>iPhone 5C</td>
<td>September 20, 2013</td>
</tr>
<tr>
<td>iPhone 5S</td>
<td>September 20, 2013</td>
</tr>
<tr>
<td>iPod touch (5th Generation)</td>
<td>October 15, 2012</td>
</tr>
<tr>
<td>iPad 2</td>
<td>March 11, 2011</td>
</tr>
<tr>
<td>iPad (3rd Generation)</td>
<td>March 16, 2012</td>
</tr>
<tr>
<td>iPad (4th Generation)</td>
<td>November 2, 2012</td>
</tr>
<tr>
<td>iPad Mini</td>
<td>November 2, 2012</td>
</tr>
<tr>
<td>iPad Mini (2nd Generation)</td>
<td>October 22, 2013</td>
</tr>
<tr>
<td>iPad Air</td>
<td>October 22, 2013</td>
</tr>
</tbody>
</table>

• To support older devices, you must specify older versions of iOS in your app through the “Deployment Target” configuration.

• For example, to support iPhone 3GS, you must target iOS version 6.1.3 or earlier.

• To support the original iPad, you must target iOS version 5.1.1 or earlier, and so on.
• **Software is written with the iOS SDK.**
  - The SDK contains all of the tools needed to create an iOS application (or “app”).
  - Both Objective-C and C are supported by the SDK.

• **Be prepared to have the development rug pulled from under you (and replaced with something better)!**
  - As the platform matures, the development environment changes ... sometimes drastically.
  - For example: with iOS 4, memory management was handled through a “Manual Retain/Release” pattern, requiring disciplined programming habits.
  - As of the release of the iOS 5 SDK, memory management is largely handled by the compiler through “Automatic Reference Counting” (ARC).
Apple Developer Membership

- **In order to write and deploy iOS apps, you must become a registered Apple Developer.**
  - Fortunately, you may “test the waters” by registering as a Free Apple Developer and viewing your app on the iOS simulator.
  - Once you are ready to start deploying your app on “real” devices, you must upgrade your account to one of the paid membership accounts.

- **There are several different types of membership:**

<table>
<thead>
<tr>
<th>Device</th>
<th>Cost</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Developer (Free)</td>
<td>Free</td>
<td>Access to iOS SDK &amp; Documentation.</td>
</tr>
<tr>
<td>iOS Developer Program (Individual)</td>
<td>$99 / Year</td>
<td>Access to Beta releases. Ability to deploy apps to “real” devices: locally, through ad hoc distribution, and through the iTunes App Store.</td>
</tr>
<tr>
<td>iOS Developer Program (Company)</td>
<td>$99 / Year</td>
<td>Same as the individual membership, but includes the ability to register multiple developers from a company at no extra charge.</td>
</tr>
<tr>
<td>iOS Enterprise Program</td>
<td>$299 / Year</td>
<td>Permits “In house” distribution of apps for employees of an enterprise. For “internal use only.” Does not permit distribution through the iTunes App Store.</td>
</tr>
<tr>
<td>iOS University Program</td>
<td>Free</td>
<td>For courses at accredited Universities.</td>
</tr>
</tbody>
</table>

- **To register as an Apple Developer (free), visit**  

  - Once you are registered, you are able to download all of the tools necessary to build iOS apps.
  - You may view your apps on the iOS Simulator.
  - In addition, you have access to print and video documentation on all things related to iOS.
• **Deploying to a “real” device requires a paid membership.**
  
  - If you run your business under your own name, register as an iOS Developer (Individual).
  
  - If you are a member of a company and are releasing apps under that company’s name, have someone with legal authority to sign binding agreements register as an iOS Developer (Company).
  
  - All participants in the iOS Developer program, whether as individuals or as part of a company, must first be registered as Apple Developers.

• **All commercial and free apps are distributed to the public through the iTunes App Store.**
  
  - Only paid and current members of the iOS Developer Program may submit apps to the iTunes App Store.

• **A single “Apple Developer” may belong to several different “companies.”**
  
  - For example, you may be registered as an individual as well as part of any number of companies with the same login/password.
  
  - When you log in to the iOS developer website or need to connect to Apple from your development environment, you are prompted with a drop-down list to choose what context you are logging in under.

• **Both individual and company iOS developer memberships are registered at:**
  
• **The Enterprise Developer program is designed for businesses that want to distribute an “in-house” app to their employees.**
  
  • Only businesses with a Dun & Bradstreet number are accepted into this program. http://www.dnb.com/
  
  • This membership prohibits distributing apps to anyone outside of the business.
  
  • If an app is distributed to people outside of the organization, whether for free or at a cost, an iOS Developer membership is required. It is permissible for an organization to have both types of membership.
  
  • Enterprise app distribution is left up to the business; the iTunes App Store may not be used with this program.
The iOS SDK

- Once you are registered as an Apple Developer, you have access to all the tools you need to write iOS apps.
  - The links for downloading these tools are found on the iOS Developer home page: https://developer.apple.com/devcenter/ios/index.action
  - There is no cost for downloading Apple’s “generally available” development tools.
  - If you wish to download “pre-release/beta” tools, you need to have either a paid account, or with some releases, you may purchase the software in the Mac App Store.

- The entire iOS SDK is included when you download Xcode.

- The iOS SDK includes:
  - Xcode – the Integrated Development Environment (IDE) for authoring iOS apps.
  - Interface Builder – a Graphical User Interface (GUI) editor, used to ease the design of visual elements for an app.
  - The iOS Simulator – a limited facsimile of an actual device (there are multiple simulators for different versions of the iPhone and iPad).
  - Instruments – A collection of tools to profile an app.
  - The iOS Application Programing Interfaces (APIs) – The code libraries needed for authoring iOS apps.
  - Documentation – Detailed explanations of how to use the included APIs.
Xcode

- **Xcode is a full-featured IDE, used to write apps for both Mac and iOS devices.**
  - Although the focus of this class is on iOS/Objective-C, Xcode is able to support other languages and APIs such as Java, C, C++, Ruby, etc.

- **Some of Xcode’s features include:**
  - Syntax coloring and highlighting.
  - Autocomplete.
  - A code “snippet” library.
  - Graphical editors with code generation.
  - Live compilation and error checking with the LLVM compiler.
  - Error messages that often include suggested fixes.
  - Debugging with configurable breakpoints.
  - Integration with Interface Builder and the iOS Simulator.
  - An assistant editor that shows related code in a separate pane.
  - Intelligent interface that auto-changes the displayed panes based on the file type you have selected or action that has occurred.
  - ... and more!
• Most of the features listed above are assembled in the Workspace Window, a collection of panes for developing apps.

• Tabs may be created to manage multiple Workspace Windows.
  • Click on the “+” sign at the far right side of the tab bar to add another tab.
The Workspace Window is broken down into four main sections:

- The Navigator Area
  - This is used to list files, display issues, search for code, and control your breakpoints.
• **The Editor Area**
  
  • All coding and design is done in this section.
• An optional Assistant Editor displays a related screen to the content in the Main Editor.

• The gutter is used to display/select breakpoints and show connections to GUI controls.
• The assistant editor can be shown or hidden by toggling the editor buttons near the top right of the workspace window.
• **The Debug Area**

  - The debug controls allow you to step through “suspended” code and jump to different threads.
  - The two panes allow you to inspect variable states, type in debug commands, and see console output.
• **The Utility Area**

  • This section is primarily used when designing screens with the “Interface Builder.”
• There are two different sections with their own selectors:

![Diagram showing different sections with their own selectors]

• Each of these sections displays different options depending on what type of code is being edited.

  • For example, the display in the Editor and Utility areas show something different when editing source code vs. graphically designing a screen.
• Each area, with the exception of the editor area, can be shown or hidden by toggling the panel buttons.

• Other features are found in the “Organizer.”
  • These include device management, project snapshots, archives and their distribution.
Interface Builder

- Interface Builder allows you to create GUIs by dragging and dropping controls, views, objects, and so on, into a workspace.
  - Interface Builder is launched simply by creating or opening a Storyboard or NIB file.
iOS APIs

- Libraries are provided by the iOS SDK to ease app development and interact with the device’s hardware.
  - The primary programming language used is Objective-C, which is a superset of C.
  - Any valid C code is also valid in Objective-C. Therefore, iOS apps typically include a mix of constructs found in both languages.
  - It is not, however, necessary to become an expert in C, as the Objective-C extensions are the primary constructs used in iOS development.

- The iOS API is organized into four logical layers (listed from highest to lowest):
  - **Cocoa Touch** – The core frameworks needed to build a GUI app for iOS.
  - **Media** – Video, audio, and animation services.
  - **Core Services** – The underlying app services, many of which are exposed in limited ways by the Cocoa Touch and Media layers.
  - **Core OS** – The lowest level device services available. These are the basic building blocks used by all of the layers above it.
  - All four layers are included in the iOS SDK.
  - When you want to use a specific library, reference the framework name, not the name of the layer. These layer names are used in documentation rather than code.
  - More on “importing” frameworks is discussed later in this course.

- Apple suggests that developers should try to stick with the higher-level APIs (Cocoa Touch and Media).
  - They are easier to write and insulate you from any potential low-level interface changes that might occur over time.
  - The higher-level layers have interfaces primarily written in Objective-C; the lower-level layers are primarily written in C.
  - Most of the work in this course is focused on frameworks found in the Cocoa Touch layer.
The iOS Simulator

- **There are simulators for both the iPhone and iPad.**
  - You can simulate a 3.5” (32 bit) or 4” (32 or 64 bit) iPhone Retina, the iPad, and iPad Retina.
  - These are the only options available for running / testing an app until you have purchased an iOS developer membership from Apple.
• The simulator has menu options and keyboard shortcuts for performing gestures, rotating the device, pressing the home and lock keys, and more.

• The simulator does not support all of the features that your app may contain (such as using the media picker or camera).

• In addition, the memory footprint and processor of an iOS device is not emulated. Real testing should be done on a real device.
Instruments

- Instruments is a powerful collection of different tools used to profile your app.

- Activities such as performance analysis, object allocation, and automated GUI testing may be executed with Instruments.
  - Profiling may be used with either the simulator or a real device (though it is recommended for most issues to use a real device).
Documentation

- All of the iOS APIs are documented and searchable.
  - Documentation is available through the “help” menu, or by holding the “option” key when you click on a statement or variable in your source code.

- Documentation is broken up into several sections:
  - **Overview** – A description of the purpose of the class and how it should be used. Links to other classes are often found in this section.
  - **Tasks** – A logical grouping of the different methods and properties available in a class. For example, methods and properties related to initializing a UITableView object are listed under a task called “Initializing a UITableView Object.”
  - Underneath the tasks are alphabetical lists of all the methods, properties, constants, and so on, for the Class.
• Make sure you are reading the right documentation for the device you are compiling to!
Putting it Together

- Four primary decisions must be made regarding the app you are building.
  - Answers to these questions determine how to configure the development environment and build targets.

1. Which devices are you targeting (iPhone, iPad, both)?
   - When you create a project in Xcode, you may choose iPhone, iPad, or Universal.
   - Don’t worry. While it’s much easier to make the correct choice up front, it’s still possible to add or change a device later.

2. What’s the lowest version of iOS you need to support?
   - This version is configured as the “Deployment Target.”

3. Which version of iOS contains the latest features you wish to include?
   - This version is configured as the “Base SDK.”
   - It may seem odd that you can specify two different versions of iOS for a single app, but Apple makes this possible through a feature called “weak linking.”
   - Weak linking, coupled with conditional coding, allows older devices to ignore features that aren’t available for their version of iOS.

4. Which peripheral features (front camera, compass, etc.) must be available to run your app?
   - “Required” features may be listed in a special properties file.
   - If the device doesn’t have the feature listed in this file, it won’t attempt to run the app.
   - Use this restriction list as a last resort. In general you should do your best to “degrade gracefully” when a feature isn’t available, rather than prevent the entire app from running.
• **iOS apps are written within an event-driven model.**
  
  • The iOS SDK provides documented “hooks” which you override with well-defined objects and methods.

  • As events occur (an app starts, a button is pushed, the device is turned on its side, the user drags his or her finger across the screen, and so on), the appropriate method is called by iOS.

  • After mastering Objective-C, most of your time learning iOS is spent discovering which protocols to conform to and which classes and methods to override.

  • Once you are familiar with the “hooks” needed for your app, develop your code in the Objective-C language, create GUIs with Interface Builder, profile your code with Instruments, and test your apps on the iOS Simulator and real devices.

• **That’s the 50,000-feet view! In the next chapter, we’ll dive into the Objective-C programming language.**
Introduction to iOS Resources


Lab Exercise: #1 - Introduction to iOS
Chapter Summary

- **iOS is the operating system for Apple’s Mobile Devices and Web TV.**

- **Apple Membership is the first step to developing iOS apps.**
  - The iOS SDK is available to all members of the free Apple Developer program.
  - In order to deploy to a real device and/or post your app to the iTunes App Store, you must enroll in Apple’s paid membership programs.

- **The iOS SDK contains everything you need to build iOS apps.**
  - Xcode
  - Interface Builder
  - Instruments
  - iOS Simulators
  - iOS APIs
  - Documentation

- **The iOS APIs allow you to interact with the features of the device through the Objective-C and C programming languages.**
  - Higher-level APIs are written in Objective-C.
  - Lower-level APIs are written in C.

- **iOS is event-driven. Most of your time coding involves conforming to well-defined Protocols, inheriting from Classes, and overriding Methods.**
  - iOS handles the low-level complexity of capturing the events and calls your code at the appropriate time.
  - This reduces the complexity and amount of code written.