Crafting iOS Apps
Course Overview

Teaching Programming with iOS

iOS, Swift and the Apple toolset provide an excellent platform for teaching students how to program. With this collection of lesson plans, instructors can engage students with a project-based curriculum, and guide students in learning object-oriented programming with Swift.

Project-Based Learning

This overview describes the structure and principles behind the projects and lesson plans for teaching a hands-on, project-based, introductory course in developing iOS apps. These course materials do not dictate a step-by-step script for your course. Rather, we provide you with a menu of projects and accompanying lessons to incorporate into your own iOS course.

We have designed the projects and lessons in this course to meet specific, clear learning outcomes, which are included in each lesson plan. Each lesson employs the need for recall and convergent, divergent, and evaluative thinking to accommodate different types of learners. These materials accommodate a wide variety of teaching styles and pedagogical approaches, including "active" learning techniques, constructionist methodology, Bloom and Blosser learning taxonomies, "Demonstrate, Collaborate, Facilitate" approaches, peer-supported learning, and your own effective methods. Lessons are flexible, and provide opportunities to answer the "what, how, what if, and why?" to engage student learning and cognition. Lastly, lessons and exercises recognize gender, ethnic and cultural bias, and we have designed these materials to be open to a diverse classroom.

Course Learning Outcomes

The primary learning outcome for this course is that students will be able to design and create iOS apps. Students will leverage Swift, the iOS SDK, and Apple developer tools. With iOS as the platform, students will learn object-oriented programming, design patterns, dynamic type systems, functional language features, message-passing paradigms, user interface design, best practices in programming, and problem analysis.

Upon successful completion of this course, students should be able to:

1. Define key programming terms relevant to Swift and iOS programming.
2. Describe the process of creating iOS apps.

3. State the purpose of the Apple developer tools, such as Xcode, Instruments, debugger, analyzer, and iOS Simulator.
4. Distinguish well-written code from poorly-written code.
5. Recognize patterns and idioms present in the Cocoa Touch API and other Apple frameworks.
6. Employ the Apple developer tools to create an iOS app.
7. Demonstrate programming best practices in Swift.
8. Examine and subdivide app functionality into properly designed components.
9. Explain and summarize iOS API features including location, mapping, sensors, gestures, multimedia and user interface components.
10. Plan, prepare and build an original iOS app, from concept to working program.

While all projects and lessons target these learning outcomes, we encourage you to pick and choose projects and lessons that best fit your own course plan.

Prerequisites

This course assumes the instructor has read a book on iOS programming, has experience creating basic iOS apps and that the instructor is comfortable with Swift. Students are expected to have prior programming experience, such as programming with Java, C++, Objective-C, Python or Ruby, in an introductory programming course. Students are not required to have prior experience with Swift.

This course assumes that students will engage in project-based, hands-on learning, either individually or in pairs, at an Apple workstation with the latest versions of OS X, Xcode and the iOS SDK installed prior to the first day of class. Lastly, using the Xcode Downloads preference item, you should have installed the latest iOS and Xcode doc sets.

Instructors can verify the development environment by running the **Level 1 Flashlight** project.

Summary of Prerequisites

- Instructor familiarity with Swift and iOS programming with Apple developer tools.
- Student familiarity with an object-oriented language such as Java, Ruby or Python.
- Instructor's Apple computer with projector, Xcode, iOS SDK and documentation installed.
- Student Apple computers with Xcode, iOS SDK and documentation installed.

Overview Of Course Materials

This curriculum is designed to teach iOS programming through the creation of multiple apps using Swift, the iOS SDK, and Apple developer tools. Each project focuses on a single app or feature set, such as a Unit Converter app that converts degrees Celsius into Fahrenheit. Projects are divided into lessons, each of which includes:

- An overview of the feature to be implemented.
- Learning outcomes and relevant vocabulary.
- Step-by-step instructions supporting instructors in leading student learning.
- Presentation slides to help communicate concepts.
- Xcode projects for instructors and students.

This menu of projects and lessons accommodates different course structures and levels of student experience. You can choose what you would like to incorporate into your course.

Levels, Projects and Lessons

All projects are grouped into levels, according to complexity. While levels are meant to be followed in sequence, we encourage you to pick and choose projects within each level according to time, student interest, course structure, and your requirements.

Level 1: Xcode Fundamentals and Swift			
WordCollage	SpaceAdventure		
Level 2: Model-View-Controller, App Fundamentals			
Clock	Stopwatch	UnitConverter	
Level 3: Frameworks and APIs			
EasyBrowser	EasyTweeter	Found	Gesturizer
FingerPainter	NoiseMaker		

Level 4: Multi-View Apps			
Flashcards	RSSReader	Journal	
Level 5: Advanced Frameworks			
ZombieSimulator			

Table 1: Projects are composed of lessons, and are grouped by levels of complexity. Instructors should pick and choose projects and lessons that meet course constraints.

Each project consists of lessons, and each lesson varies in content and difficulty. Rather than "squeezing in" an hour or two's worth of material, a lesson focuses on a particular app feature, describes how to implement the feature, and emphasizes the concepts necessary for understanding what has been done. You may find that some class meetings will consist of two lessons, or that a lesson may need to span multiple meetings.

	M	T	W	R	F	M	T	W	R	F
UnitConverter Lesson	1,2	3	4	4,5	6	7	8	8,9	10	11

Table 2: Not all lessons are equal in length. Multiple lessons may fit in one class meeting, while others may require two class meetings.

Which projects and lessons to incorporate, and how to best fit them into your specific course, are up to you. For example, after an introductory sequence with Swift, you may wish to start with building the Clock app, which includes six lesson plans, concept slides and Xcode projects. Next, you might build the UnitConverter app, which includes twelve lesson plans, slides and Xcode projects. You might then move on to the next level, and build EasyTweeter and NoiseMaker, each with their accompanying lessons, slides and code.

Level 1			
Word Collage	SpaceAdventure		
Level 2			
Clock	Stopwatch	UnitConverter	

Level 3			
EasyBrowser	EasyTweeter	Found	Gesturizer
FingerPainter	NoiseMaker		
Level 4			
Flashcards	RSSReader	Journal	
Level 5			
ZombieSimulator			

Table 3: Instructors may treat the levels and projects as a menu, to meet student interest, experience, and course constraints.

Although the projects vary in complexity, we have designed each project's lessons to reach the most important learning outcomes as early as possible. This provides instructors and students the option to work on a different app each week, and yet reach the most important learning outcomes, despite not finishing the later lessons within a single project. Rather than spend two weeks on a single project's lessons, instructors may choose to move on to another project, to maintain student interest.

Week	Project Lessons	Concepts	Level
1	WordCollage, SpaceAdventure	Developer Tools, Swift	1
2	SpaceAdventure	Swift	
3	Clock	Views, contollers, and outlets	2
4	StopWatch	MVC, actions, outlets, and NSTimer	
5	UnitConverter	UI Components, Protocols, Delegates	
6	EasyBrowser	WebKit Framework	3
7	EasyTweeter	Social Framework	

Week	Project Lessons	Concepts	Level
8	Found	Map Kit Framework	3
9	Gesturizer	UIGestureRecognizer	
10	FingerPainter	UITouch, CoreGraphics	
11	NoiseMaker	AVFoundation Framework	
12	Flashcards	Multiple Views and Segues	4
13	RSSReader	Tab Bar Controllers, Navigation Controllers	
14	Journal	Table View Controllers, Core Data	
15	Zombie Simulator	Sprite Kit	5
16	Zombie Simulator	Sprite Kit	

Table 4: A weekly plan for a semester that focuses on one project per week.

Engaging Advanced Students

Many lessons include an **Extensions and Modifications** supplement to support instructors in keeping advanced students engaged in exploring iOS development. Here are some additional suggestions we have found effective and challenging for advanced students.

- **Leverage them as assistants.** When an advanced student finishes an exercise early, empower them as assistants to facilitate other students' learning.
- **Ask them to explain technical details.** Much room is left in this curriculum to explore programming topics deeply. Ask advanced students to explain technical details or encourage them to seek out the answers to technical questions online.
- **Present harder problems.** While the "Extensions" sections provide additional work related to the lesson, challenge advanced students with genuinely hard problems.
- **Encourage robustness, better error handling and emphasize user experience.** Instruct them to explore Instruments, to become more adept with the debugger, to consider and handle edge cases and to think deeper about user experience. (What makes a great app?)
- **Encourage them to create another app of their own choosing.** Inform them that they must solve their own problems and bugs although you will try to facilitate.

- **Elevate their workflow to a professional level.** Encourage the advanced student to spend time with the Apple developer documentation, to use version control and to explore libraries and frameworks.
- **Deploy to the device.** Enable the advanced student to deploy their work to an actual iOS device to explore the sensors and camera.

Education Standards

The lessons and exercises acknowledge and support the standards of the ACM Computer Science Teachers Association and the College Board AP Computer Science standards.

CSTA Project Based, Level 3 Course

- Computational thinking
- Collaboration
- Computing Practice & Programming
- Computer & Communications Devices
- Community, Global and Ethical Impacts

AP Computer Science Criteria

- Object-Oriented Program Design
- Program Implementation
- Program Analysis
- Standard Data Structures
- Computing in Context