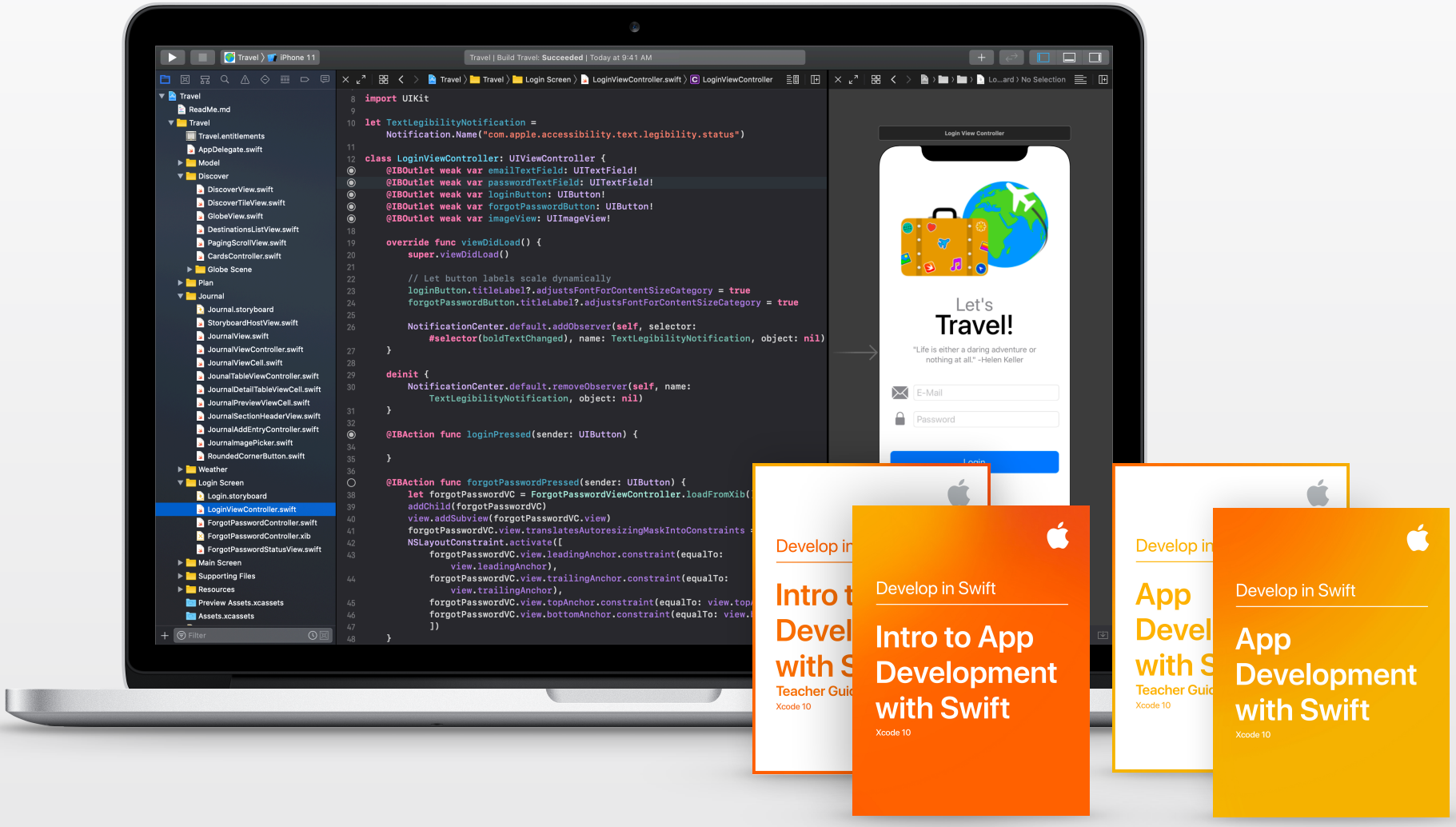
Develop in Swift

Curriculum Guide



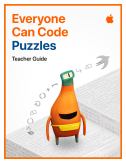
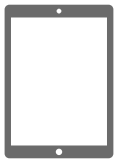
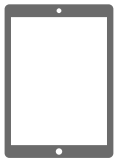
November 2019

Teaching Code with Apple

When you teach code, you're not only teaching the language of technology. You're teaching new ways to think and bring ideas to life. And coding with Swift, Apple's powerful, intuitive and easy-to-learn programming language, provides students fun and engaging ways to prepare for the future. Every student should have the opportunity to create something that can change the world. Whether students are just getting started with Swift Playgrounds on iPad or ready to learn Xcode on Mac, Apple provides everything educators need to bring code into the classroom.

Teaching Code Curriculum Pathway

The Everyone Can Code and Develop in Swift curricula take students all the way from writing their first lines of Swift code to building their first apps. The table below provides an overview of the core, free teaching and learning resources available.

	Student	Teacher	Device	Audience	App	Prerequisites	Overview	Lesson hours
Everyone Can Code								
Everyone Can Code Puzzles				Year 5 and up	Swift Playgrounds 	None	Students learn foundational coding concepts, such as loops, variables and functions. They practise and apply their skills in a variety of ways, from puzzles to open-ended playgrounds.	45 hours
Everyone Can Code Adventures				Year 5 and up	Swift Playgrounds 	Everyone Can Code Puzzles	Students learn more advanced code concepts, such as event handling, advanced arrays and component-based design as they scope and build their own projects.	45 hours
Develop in Swift								
Intro to App Development with Swift				Secondary school and up	Xcode 	None	Students get practical experience with the tools, techniques and concepts needed to build a basic iOS app from scratch.	90 hours
App Development with Swift				Secondary school and up	Xcode 	None	Students build a foundation in Swift, UIKit and networking through hands-on labs and guided projects. By the end of the course, students can build an app of their own design.	180 hours

Develop in Swift

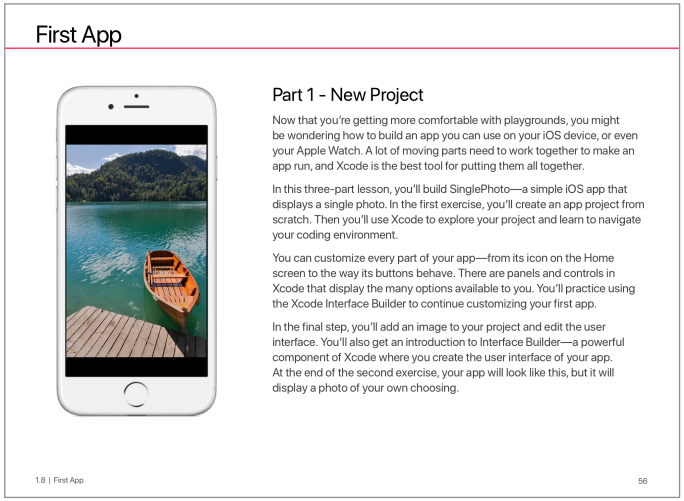
Creating apps in Swift gives students a way to think about using code to make an impact in the world. The Develop in Swift curriculum teaches students Swift, a powerful and intuitive open source programming language included with the Xcode programming environment on Mac. It's the same language professional developers use in the fast-growing app economy to make apps for iOS, macOS, tvOS, watchOS and beyond. Develop in Swift is great for teaching students who are new to coding as well as for advancing those with previous experience. It even prepares them for a career in programming with industry-recognised certification.

Curriculum Overview

Develop in Swift is intended for secondary school and higher education students to explore designing and building a fully functioning app of their own. As they develop new skills, students master key coding concepts and can even earn an industry-recognised certification for knowledge of Swift and Xcode. Students with certification receive a digital badge that they can share on professional networks to convey their preparedness for industry. Teacher guides are available for educators – regardless of experience teaching Swift or other programming languages – that provide tools for deepening engagement with aspiring app developers. For after-school or summer learning programmes, there are complementary Swift Coding Club materials. And app showcases give coders a chance to celebrate their ingenuity with the community, whether they're learning in or out of the classroom.

The Intro to App Development with Swift course introduces students to the world of app development and the basics of Swift and Xcode. The course culminates in a final project where they can choose one of two basic iOS apps to build.

App Development with Swift follows Intro to App Development and offers students the chance to go further. Students who are familiar with Swift, Xcode and iOS development can move through lessons quickly or go straight to the labs, where they'll build mini-projects and test their code in playgrounds. By the end of the course, they'll be able to build a fully functioning app of their own design. Once students complete this course, they can earn industry-recognised certification for knowledge of Swift and Xcode.



First App

Part 1 - New Project

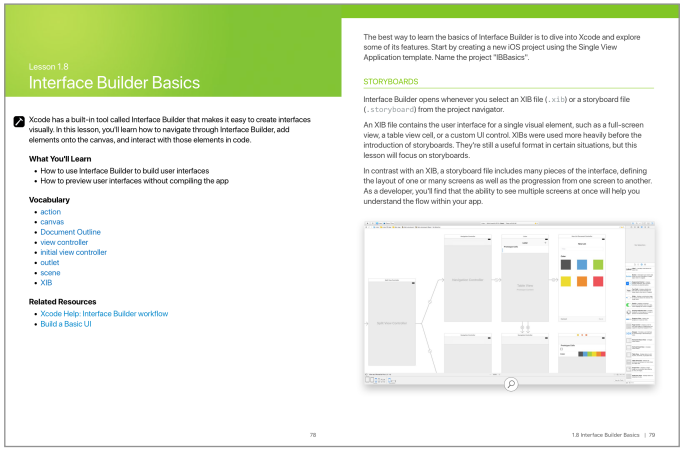
Now that you're getting more comfortable with playgrounds, you might be wondering how to build an app you can use on your iOS device, or even your Apple Watch. A lot of moving parts need to work together to make an app run, and Xcode is the best tool for putting them all together.

In this three-part lesson, you'll build SinglePhoto—a simple iOS app that displays a single photo. In the first exercise, you'll create an app project from scratch. Then you'll use Xcode to explore your project and learn to navigate your coding environment.

You can customize every part of your app—from its icon on the Home screen to the way its buttons behave. There are panels and controls in Xcode that display the many options available to you. You'll practice using the Xcode Interface Builder to continue customizing your first app.

In the final step, you'll add an image to your project and edit the user interface. You'll also get an introduction to Interface Builder—a powerful component of Xcode where you create the user interface of your app. At the end of the second exercise, your app will look like this, but it will display a photo of your own choosing.

1.8 | First App 56



Lesson 1.8

Interface Builder Basics

Xcode has a built-in tool called Interface Builder that makes it easy to create interfaces visually. In this lesson, you'll learn how to navigate through Interface Builder, add elements onto the canvas, and interact with those elements in code.

What You'll Learn

- How to use Interface Builder to build user interfaces
- How to preview user interfaces without compiling the app

Vocabulary

- action
- canvas
- Document Outline
- view controller
- initial view controller
- outlet
- scene
- XIB

Related Resources

- Xcode Help: Interface Builder workflow
- Build a Basic UI

The best way to learn the basics of Interface Builder is to dive into Xcode and explore some of its features. Start by creating a new iOS project using the Single View Application template. Name the project "IBBasics".

STORYBOARDS

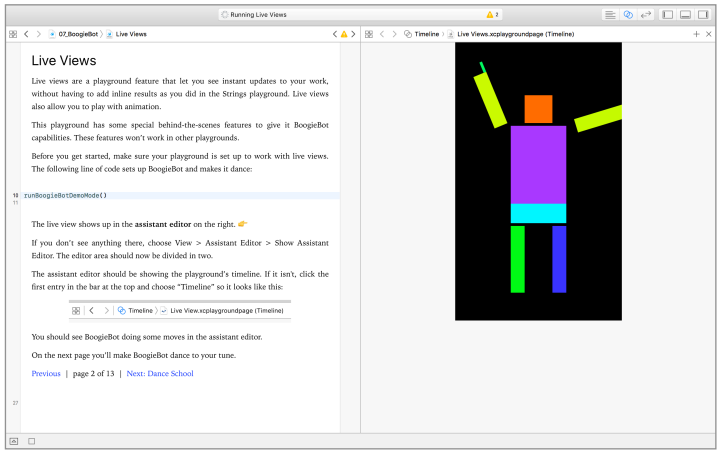
Interface Builder opens whenever you select an XIB file (`.xib`) or a storyboard file (`.storyboard`) from the project navigator.

An XIB file contains the user interface for a single visual element, such as a full-screen view, a table view cell, or a custom UI control. XIBs were used more heavily before the introduction of storyboards. They're still a useful format in certain situations, but this lesson will focus on storyboards.

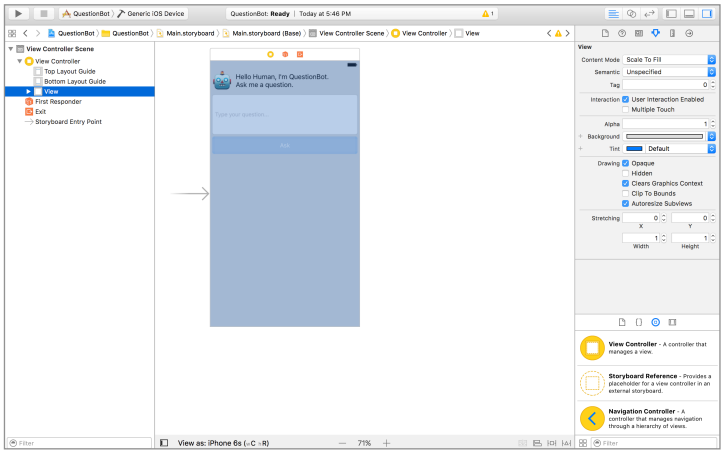
In contrast with an XIB, a storyboard file includes many pieces of the interface, defining the layout of one or many screens as well as the progression from one screen to another. As a developer, you'll find that the ability to see multiple screens at once will help you understand the flow within your app.

78 1.8 Interface Builder Basics | 79

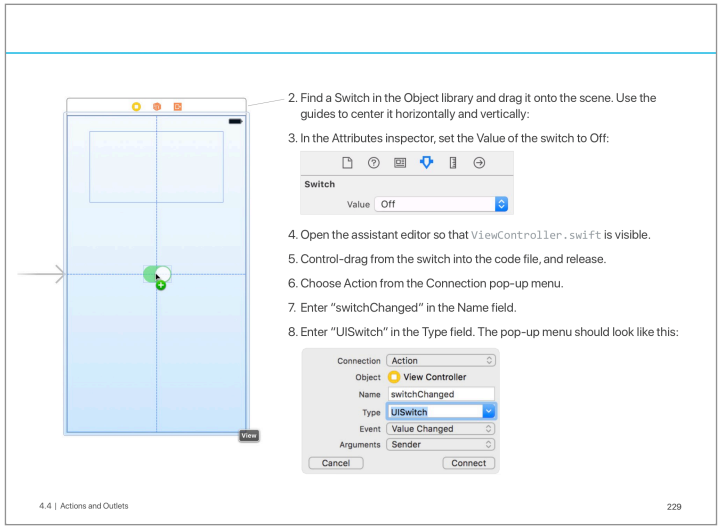
Key Features



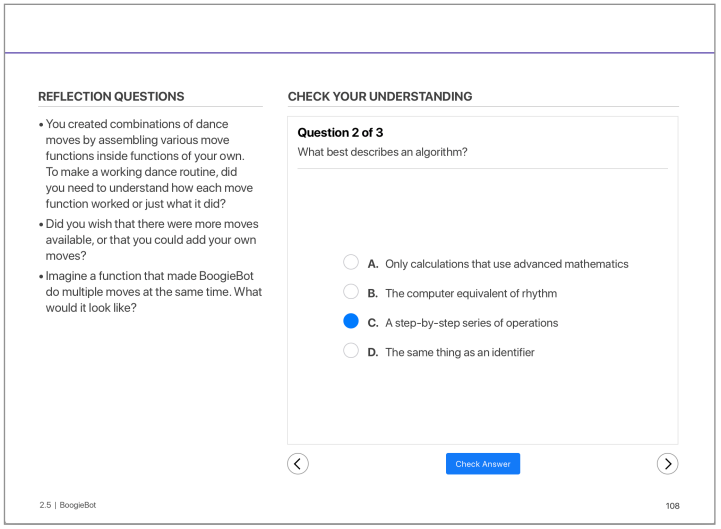
Xcode playgrounds. Students learn programming concepts as they write code in Xcode playgrounds – interactive coding environments that let them experiment with code and see the results immediately.



Sample projects. Using the included project files, students can work with parts of code without having to build an entire app from the beginning.



Step-by-step instructions. Detailed instructions with images and videos guide students through all the steps of building an app in Xcode.



Study tools. Students can check their understanding and reflect on what they've learned with review questions, key vocabulary, links to documentation and more.

Develop in Swift Curriculum

Intro to App Development with Swift Course Outline

This introductory one-semester/90-hour course is designed to help students build a solid foundation in programming fundamentals using Swift as the programming language. Students get practical experience with the tools, techniques and concepts needed to build a basic iOS app. App design lessons take students through the app design process, which includes brainstorming, planning, prototyping and evaluating an app of their own. While students may not yet have the skills to actually build an app, the work they put into their app prototypes sets them up for future development.

Lesson 1: Playground Basics. Students become familiar with the interactive playground environment.

Lesson 2: Naming and Identifiers. Students explore the fundamentals of solving problems by using good names and identifiers.

Lesson 3: Strings. Students are introduced to the concept of strings and string interpolation.

Lesson 4: Hello, world! Students are welcomed to the tradition of programming, learning how to customise their Xcode environment and debugging.

Lesson 5: First App. Students create their first app using Xcode, displaying their work in an iOS simulator.

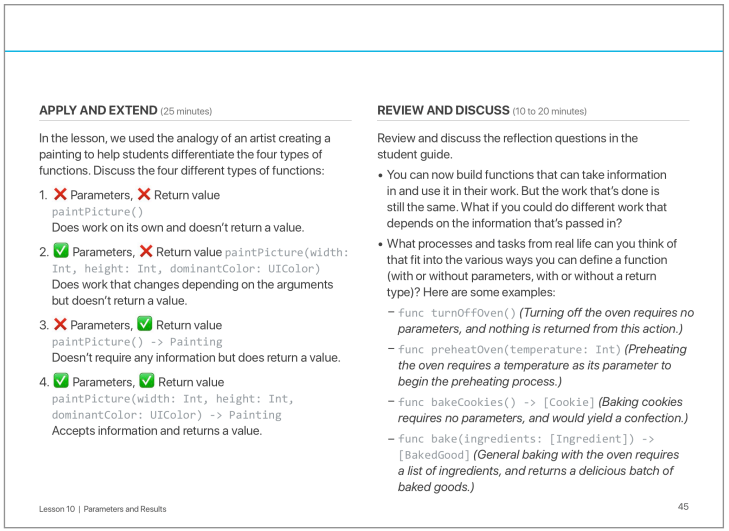
Lesson 6: Functions. Students discover what makes functions so powerful as they combine detailed steps into a definition they can use again and again.

Lesson 7: BoogieBot. Students put their knowledge of functions to work by controlling an animated dancing robot within the playground.

Lesson 8: Constants and Variables. Students expand their understanding of naming through an introduction to the concepts of constants and variables.

Lesson 9: Types. Students become more familiar with the foundation of Swift by examining the type system – from types in the standard Swift library to custom types.

Lesson 10: Parameters and Results. Students expand their knowledge of functions by finding out about parameters and return values and how they make functions more flexible and powerful.



APPLY AND EXTEND (25 minutes)

In the lesson, we used the analogy of an artist creating a painting to help students differentiate the four types of functions. Discuss the four different types of functions:

1. ❌ Parameters, ❌ Return value
   `paintPicture()`
   Does work on its own and doesn't return a value.
2. ✅ Parameters, ❌ Return value `paintPicture(width: Int, height: Int, dominantColor: UIColor)`
   Does work that changes depending on the arguments but doesn't return a value.
3. ❌ Parameters, ✅ Return value
   `paintPicture() -> Painting`
   Doesn't require any information but does return a value.
4. ✅ Parameters, ✅ Return value
   `paintPicture(width: Int, height: Int, dominantColor: UIColor) -> Painting`
   Accepts information and returns a value.

REVIEW AND DISCUSS (10 to 20 minutes)

Review and discuss the reflection questions in the student guide.

- You can now build functions that can take information in and use it in their work. But the work that's done is still the same. What if you could do different work that depends on the information that's passed in?
- What processes and tasks from real life can you think of that fit into the various ways you can define a function (with or without parameters, with or without a return type)? Here are some examples:
  - `func turnOffOven()` *(Turning off the oven requires no parameters, and nothing is returned from this action.)*
  - `func preheatOven(temperature: Int)` *(Preheating the oven requires a temperature as its parameter to begin the preheating process.)*
  - `func bakeCookies() -> [Cookie]` *(Baking cookies requires no parameters, and would yield a confection.)*
  - `func bake(ingredients: [Ingredient]) -> [BakedGood]` *(General baking with the oven requires a list of ingredients, and returns a delicious batch of baked goods.)*

Lesson 10 | Parameters and Results

45

The teacher guide includes additional extension activities, discussion questions and activities for the App Design Journal that students maintain throughout the course.

Intro to App Development Course Outline (continued)

Lesson 11: Making Decisions. Students learn how to make decisions in code using conditional if/else statements, true or false Bool values and comparison operators.

Lesson 12: Instances, Methods and Properties. Students build on their knowledge of types by exploring the methods and properties that make up an instance of that type.

Lesson 13: QuestionBot. Students get experience modifying an existing Xcode project by writing new logic for an app bot that responds to different questions.

Lesson 14: Arrays and Loops. Students discover how to create and work with arrays by adding and removing objects. They also learn how loops work with each object in an array.

Lesson 15: Defining Structures. Students recognise that it's often useful to group related information and functionality into a custom type.

Lesson 16: QuestionBot 2. Students expand on the QuestionBot app by building ChatBot, an app that displays the history of a conversation. They examine the data source pattern, and build a simple data source object to provide information on Message objects to display in the message list view. Students practise appending to an array to store messages on the data source object to maintain a history of the conversation.

Lesson 17: Actions and Outlets. Students find out how to a build user interface (UI) with Interface Builder and connect UI elements with code through outlets and actions. They practise creating outlets to access properties of a UI view and creating actions to respond to user interaction with buttons and other controls.

Lesson 18: Adaptive User Interfaces. Students learn a repeatable process to create a UI on the smallest-size iPhone that scales up to all iPhone sizes and orientations. They explore Auto Layout, the system for laying out constraints that set the location and size of UI elements. And they use stack views, a special object designed to automatically set auto layout constraints based on simpler settings and a grid-like system. In the process, they build the SimpleCenter, ElementQuiz and AnimalSounds apps.

Lesson 19: Enumerations and Switch. Students discover enumerations (or enums) – a way to define a named list of options. They learn what they're used for, how to define them, and common ways to work with them. They also learn to use the switch statement to conditionally run specific code based on any option that an enum defines.

Lesson 20: Final Project. Students complete one or both final project options from scratch. The first option is a Rock/Paper/Scissors game; the second is a meme generator. Students review a variety of concepts covered in the course and build the user interface, model data and controller objects that make up the entire app.

Lesson 21: App Design. Students go through a design cycle that focuses on prototyping, much like the process that professional app developers go through.

App Development with Swift Course Outline

This two-semester/180-hour course features 51 lessons, each designed to teach a specific skill related to either Swift or app development. At the end of each of the first five units, students complete guided projects that include a description of user-centred features, a project plan and instructions for designing and building a fully functioning app. Through these projects, students can create features that interest them, all while performing the type of work they can expect in an app development workplace.

Unit 1: Getting Started with App Development. Students find out about the basics of data, operators and control flow in Swift as well as documentation, debugging, Xcode, building and running an app, and Interface Builder. They then apply this knowledge to a guided project called Light, in which they create a simple torch app.

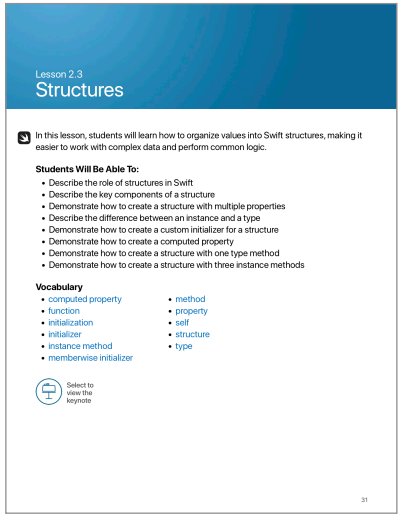
Unit 2: Introduction to UIKit. Students explore Swift strings, functions, structures, collections and loops. They also learn about UIKit — the system views and controls that make up a user interface — and how to display data using Auto Layout and stack views. They put this knowledge to practise in a guided project called Apple Pie, where they build a word-guessing game app.

Unit 3: Navigation and Workflows. Students discover how to build simple workflows and navigation hierarchies using navigation controllers, tab bar controllers and segues. They also examine two powerful tools in Swift, optionals and enumerations. They put this knowledge into practice with a guided project called Personality Quiz, a personalised survey that reveals a fun response to the user.

Unit 4: Tables and Persistence. Students find out about scroll views, table views and building complex input screens. They also explore how to save data, share data to other apps and work with images in a user's photo library. They use their new skills in a guided project called List, a task-tracking app that allows the user to add, edit and delete items in a familiar table-based interface. Students can customise the app to keep track of all information types, such as collections, tasks or playlists.

Unit 5: Working with the Web. Students learn about animations, concurrency and working with the web. They apply what they've learned in the guided project called Restaurant, a customisable menu app that displays the available dishes from a restaurant and allows the user to submit an order. The app uses a web service that lets students set up the menu with their own menu items and photos.

Unit 6: Prototyping and Project Planning. Students learn how to design, prototype and architect a project of their own design. Given enough time, they should be able to build this project independently.



Lesson 2.3
Structures

In this lesson, students will learn how to organize values into Swift structures, making it easier to work with complex data and perform common logic.

Students Will Be Able To:

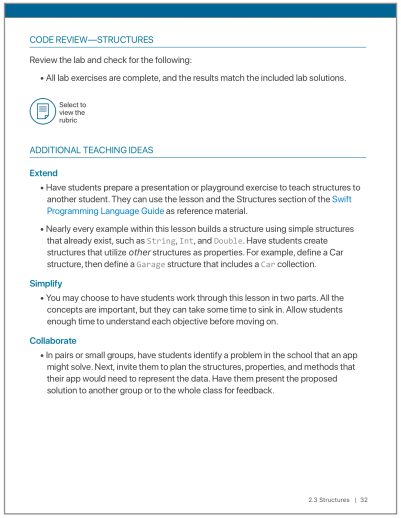
- Describe the role of structures in Swift
- Describe the key components of a structure
- Demonstrate how to create a structure with multiple properties
- Describe the difference between an instance and a type
- Demonstrate how to create a custom initializer for a structure
- Demonstrate how to create a computed property
- Demonstrate how to create a structure with one type method
- Demonstrate how to create a structure with three instance methods

Vocabulary

- computed property
- function
- initialization
- initializer
- instance method
- memberwise initializer
- method
- property
- self
- structure
- type

Select to view the keynote

31



CODE REVIEW—STRUCTURES

Review the lab and check for the following:

- All lab exercises are complete, and the results match the included lab solutions.

Select to view the rubric

ADDITIONAL TEACHING IDEAS

Extend

- Have students prepare a presentation or playground exercise to teach structures to another student. They can use the lesson and the Structures section of the Swift Programming Language Guide as reference material.
- Nearly every example within this lesson builds a structure using simple structures that already exist, such as `String`, `Int`, and `Double`. Have students create structures that utilize *other* structures as properties. For example, define a Car structure, then define a `Garage` structure that includes a `Car` collection.

Simplify

- You may choose to have students work through this lesson in two parts. All the concepts are important, but they can take some time to sink in. Allow students enough time to understand each objective before moving on.

Collaborate

- In pairs or small groups, have students identify a problem in the school that an app might solve. Next, invite them to plan the structures, properties, and methods that their app would need to represent the data. Have them present the proposed solution to another group or to the whole class for feedback.

2.3 Structures | 32

The teacher guide includes tips for extending or adapting lessons and for supporting students who need additional assistance.

Develop in Swift Support Materials

Assessment

Each teacher guide in Develop in Swift includes project-based assessments that allow educators to observe students' collaboration, communication and critical-thinking skills and to assess their coding and documentation through submitted tasks.

Certification

Educators teaching App Development with Swift can help students earn recognition for their knowledge of Swift and Xcode. App Development with Swift Level 1 certification is available through an exam administered by Certiport,* and shows that students are ready to take the next step in becoming app developers. Learn more about certification >

App Design Journal

Even before they can begin to develop apps in Swift, students can think about the types of apps they might want to design and build. The App Design Journal guides students through prototyping their ideas, testing the app with their peers and refining the user experience.

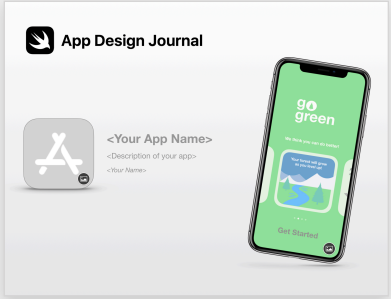
App Showcase Guide

Celebrate student ingenuity. Encourage students to share their coding achievements with the broader community through community events, such as project demonstration events or app showcases. The App Showcase Guide provides practical support to help you plan and execute a showcase event.

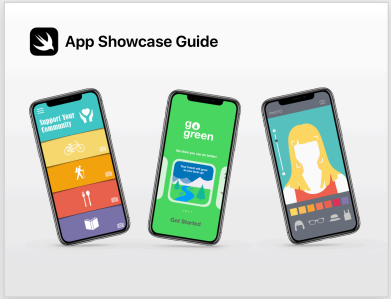
Swift Coding Club

Engage and extend students beyond the classroom with a Swift Coding Club. Fun activities let students collaborate in challenge-based activities, from programming connected devices to designing their own apps. Facilitator guides give any club leader the tools to get students creating with code.

*Additional terms may apply; see the Certiport website for more information.



Download the App Design Journal >



Download the App Showcase Guide >

Explore the Swift Coding Club >

Teaching Code Additional Information

Download the Swift Playgrounds resources

- Everyone Can Code Puzzles
- Everyone Can Code Puzzles Teacher Guide
- Swift Playgrounds app

Download the App Development with Swift guides

- Intro to App Development with Swift
- Intro to App Development with Swift: Teacher Guide
- App Development with Swift
- App Development with Swift: Teacher Guide

Additional resources

- Learn more about Apple's programs for teaching code.
- Connect with other educators in the Apple Developer Forums.
- Get App Development with Swift Level 1 certification.
- Import App Development with Swift into your Canvas instance.

About Swift

Swift is the powerful and intuitive programming language created by Apple for building apps. Swift is not only great for getting you started with coding, it's also super powerful. It's designed to scale from writing the simplest program, like "Hello, world!", to the world's most advanced software. Learn more about Swift.

About Xcode

Xcode is the Mac app used to build every other Mac app and every iOS app too. It has all the tools you need to create an amazing app experience. And it's available as a free download from the Mac App Store. Learn more about Xcode.

Apple Professional Learning

Apple Professional Learning Specialist offerings are organised into multiple-day engagements over a period of time and are designed to provide:

- Leadership visioning and planning to help administrators prioritise learning objectives.
- Guidance from an Apple Professional Learning Specialist who's dedicated to your project.
- Customised, research-based professional learning plans to match learning goals.
- Hands-on, immersive learning experiences to help faculty develop innovative instructional practices that engage students.
- An opportunity to work with your professional learning leaders to build sustainability.

Let Apple Professional Learning Specialists help provide your faculty with the best personal support for innovating their instructional practices in ways that will engage students and enable personalised learning.

To learn more, contact your Apple account executive or send an email to apls@apple.com.

November 2019