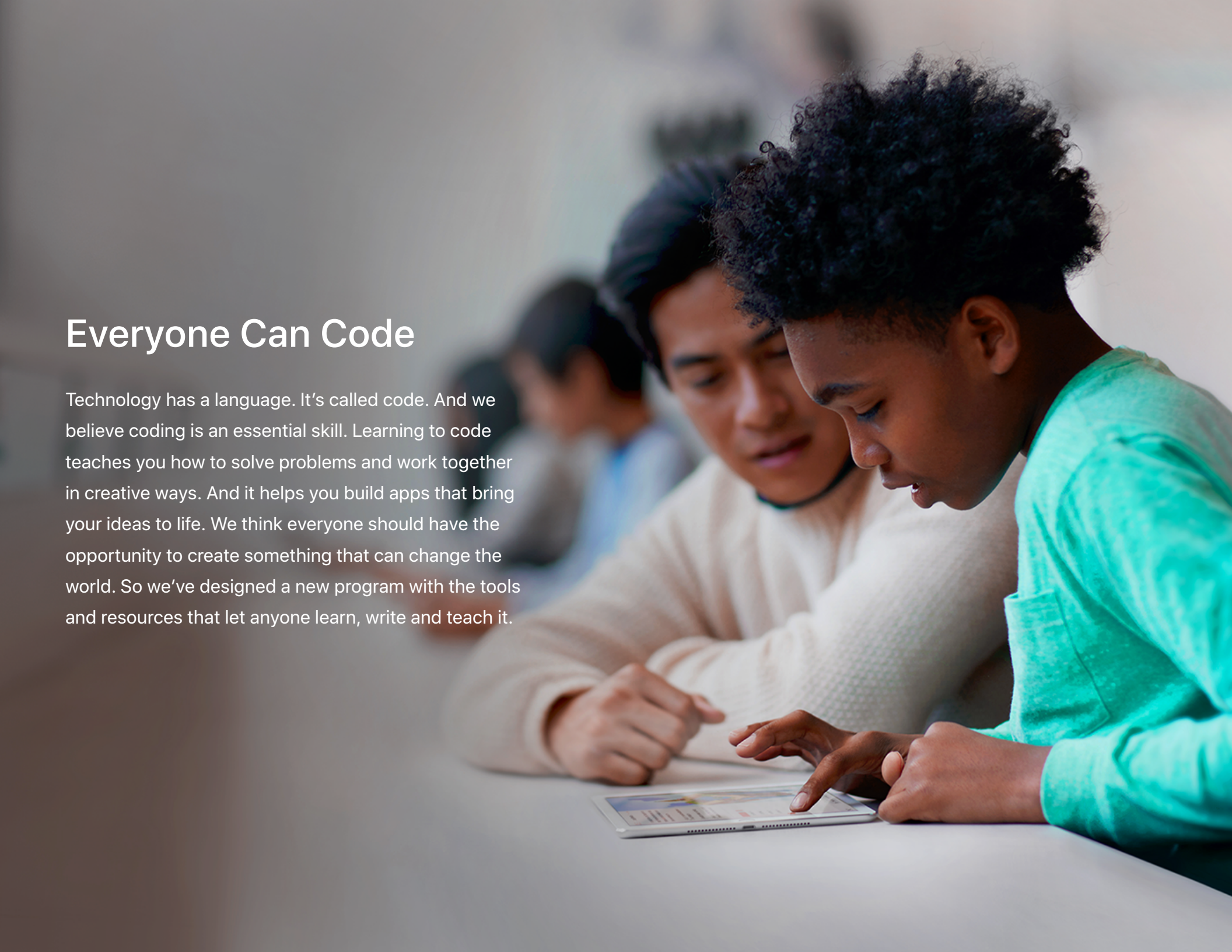
App Development with Swift Curriculum Guide

October 2017

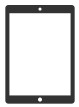
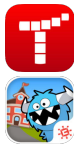
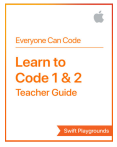
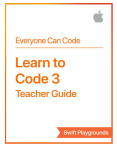
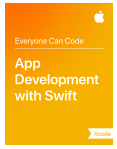
Everyone Can Code

Technology has a language. It's called code. And we believe coding is an essential skill. Learning to code teaches you how to solve problems and work together in creative ways. And it helps you build apps that bring your ideas to life. We think everyone should have the opportunity to create something that can change the world. So we've designed a new program with the tools and resources that let anyone learn, write and teach it.



Everyone Can Code Curriculum

The Everyone Can Code program includes a range of resources that take students all the way from no coding experience to building their first apps. The table below provides an overview of all the free teaching and learning resources available.

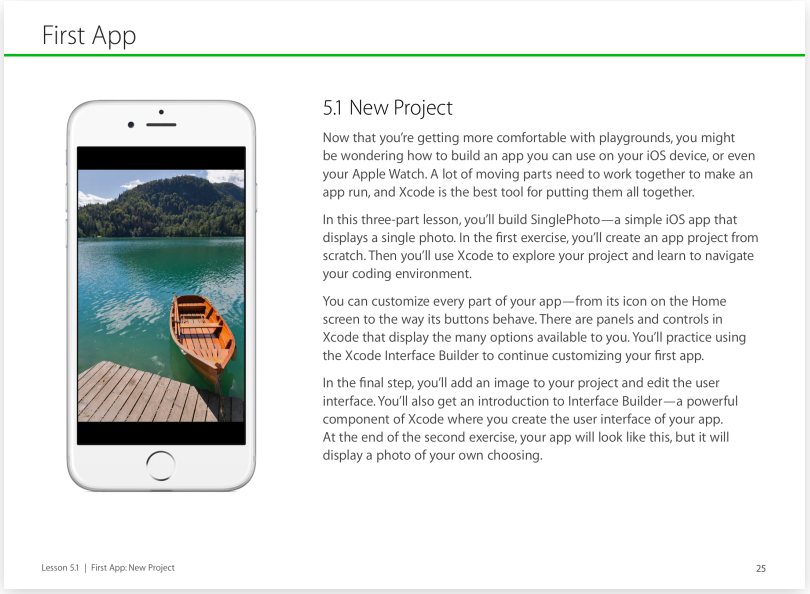
Curriculum	Device	Audience	App	Prerequisites	Overview	Learning materials	Support resources	Number of lesson hours included
		Years K through 2		None	Begin to think like coders with hands-on explorations of coding concepts using visual-based apps.	• codeSpark Academy app lessons • Tynker Space Cadet course	• *Get Started with Code 1 Teacher Guide*	30 hours, including Teacher Guide and app lessons
		Years 3 through 5		None	Explore fundamental coding concepts, and practise thinking like coders using visual-based apps.	• Tynker Dragon Spells course	• *Get Started with Code 2 Teacher Guide*	36 hours, including Teacher Guide and app lessons
		Years 6 through 10		None	Learn fundamental coding concepts using real Swift code.	• Swift Playgrounds app • Learn to Code 1 & 2 lessons • iTunes U course	• *Learn to Code 1 & 2 Teacher Guide* • Apple Teacher Learning Center Swift Playgrounds badges	Up to 85 hours, including Teacher Guide and Learn to Code 1 & 2 lessons
		Years 6 through 10		Learn to Code 1 & 2	Expand coding skills and start thinking more like an app developer.	• Swift Playgrounds app • Learn to Code 3 lessons	• *Learn to Code 3 Teacher Guide*	Up to 45 hours, including Teacher Guide and Learn to Code 3 lessons
		Senior high school and university		None	Get practical experience with the tools, techniques and concepts needed to build a basic iOS app from scratch.	*Intro to App Development with Swift* book and project files	• *Intro to App Development with Swift Teacher Guide*	90 hours
		Senior high school and university		None	Build a foundation in Swift, UIKit and networking through hands-on labs and guided projects. Students can build an app of their own design by the end of the course.	*App Development with Swift* book and project files	• *App Development with Swift Teacher Guide*	180 hours

Overview

The Intro to App Development with Swift and App Development with Swift curriculums were designed to teach high school and university students with little or no programming experience how to be app developers, capable of bringing their own ideas to life.

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 students can choose one of two basic iOS apps to build.

App Development with Swift takes students further, whether they're new to coding or want to expand their skills. If they're already familiar with Swift, Xcode and iOS development, they 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.



First App

5.1 New Project

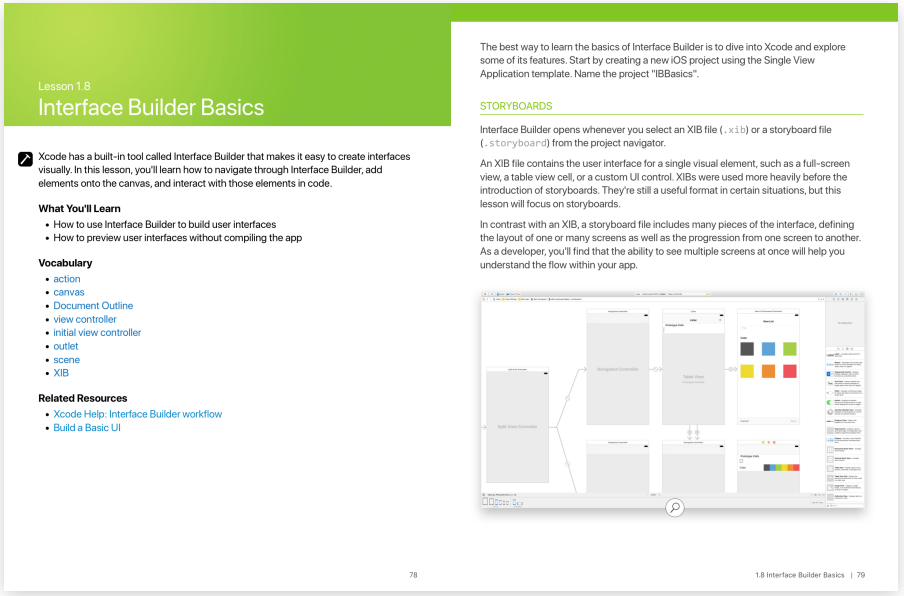
Now that you're getting more comfortable with playgrounds, you might be wondering how to build an app you can use on your iOS device, or even your Apple Watch. A lot of moving parts need to work together to make an app run, and Xcode is the best tool for putting them all together.

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.

Lesson 5.1 | First App: New Project 25



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.

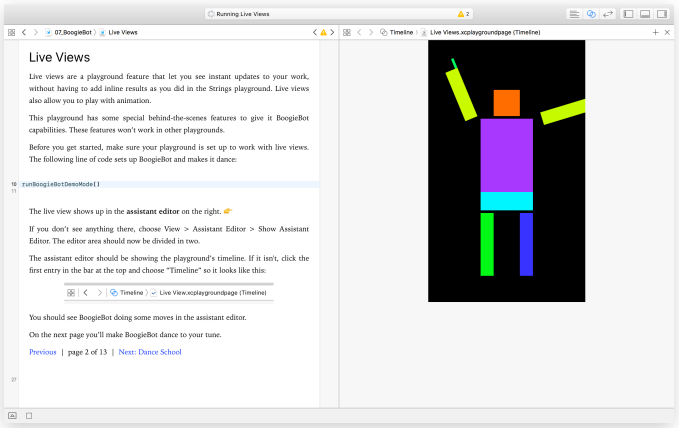
An XIB file contains the user interface for a single visual element, such as a full-screen view, a table view cell, or a custom UI control. XIBs were used more heavily before the introduction of storyboards. They're still a useful format in certain situations, but this lesson will focus on storyboards.

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.

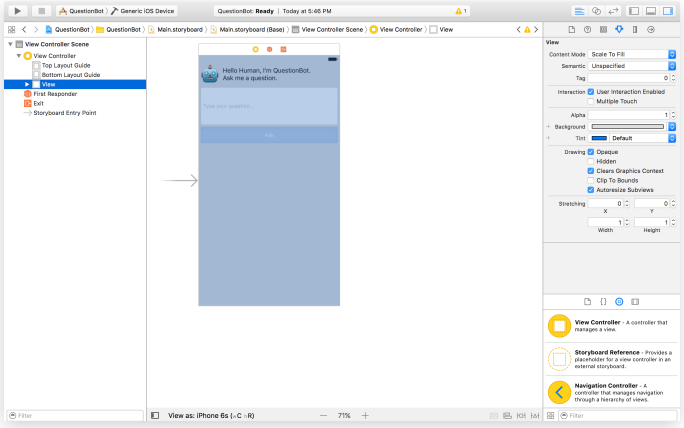
78 1.8 Interface Builder Basics | 79

Key Features

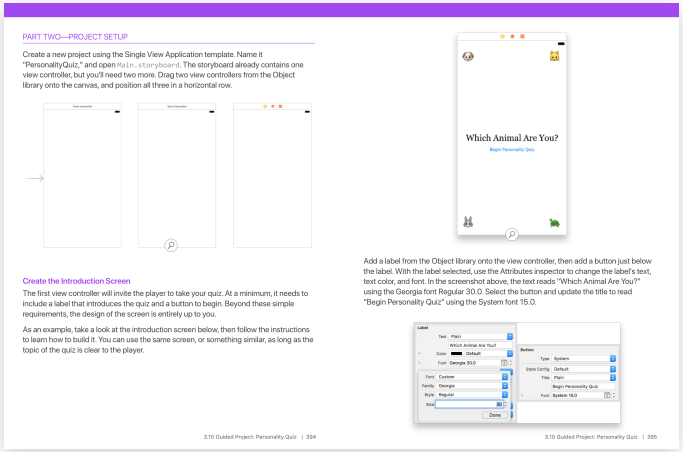
Playgrounds. Students learn programming concepts as they write code in playgrounds — an interactive coding environment that lets them experiment with code and see the results immediately.



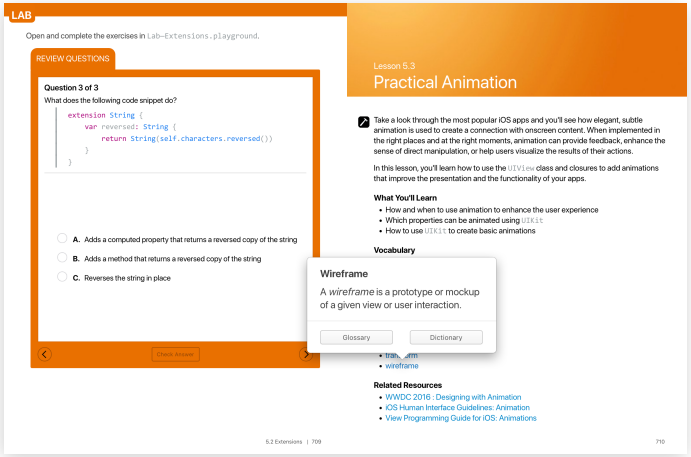
Sample projects. Using the included project files, students can try out certain parts of code without having to build an entire app from the beginning.



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 apply what they've learned with review questions, key vocabulary, links to documentation and more.



Course Outlines

Intro to App Development with Swift

This introductory one-semester course is designed to help students build a solid foundation in programming fundamentals using the Swift 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 process of designing an app, including brainstorming, planning, prototyping and evaluating an app of their own. Even though they might not yet have the skills to build the app, the work they put into the prototype will set them up for future development.

Lesson 1 — Playground Basics. Students get 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 to debug.

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.

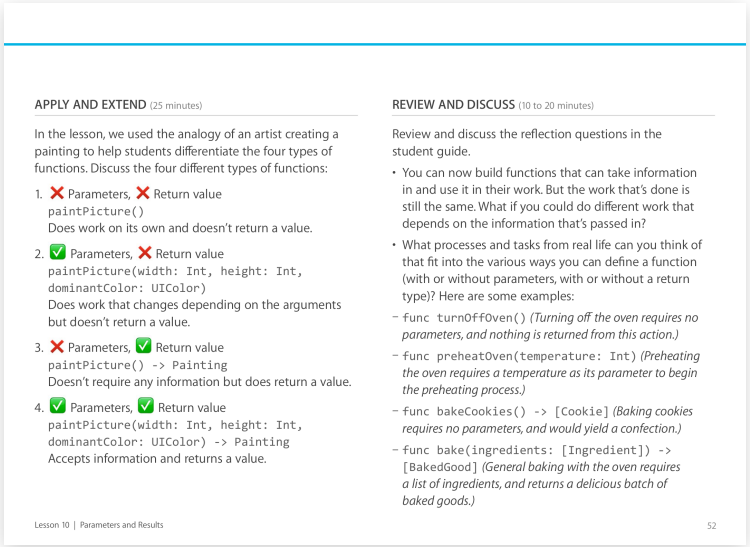
Lesson 7: — BoogieBot. Students put their knowledge of functions to work by controlling an animated dancing robot within the playground.

Lesson 8 — Constants and Variables. Students expand their understanding of naming as they're formally introduced to the concepts of constants and variables.

Lesson 9 — Types. Students become more familiar with the underpinnings of Swift by examining the type system, from the standard library in Swift to customised types.

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

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.



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 52

The Teacher Guide includes additional extension activities, discussion questions and activities for the app journal that students maintain throughout the course.

Course Outlines (continued)

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, and how for-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 customised type.

Lesson 16 — QuestionBot 2. Students expand on the QuestionBot app by building ChatBot, an app that displays the history of the conversation. They'll 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 and maintain a history of the conversation.

Lesson 17 — Actions and Outlets. Students find out how to build user interfaces using Interface Builder, and tie user interface elements into code via Outlets and Actions. They'll practise creating Outlets to access properties of a user interface view, and Actions to respond to user interaction with buttons and other controls.

Lesson 18 — Adaptive User Interfaces. Students learn a repeatable process to create a user interface on the smallest iPhone device size that scales up to all iPhone device sizes and orientations. They'll explore Auto Layout, the system for laying out constraints that set the location and size of user interface elements. And they'll 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 that enumerations, or enums, are a way to define a named list of options, what they're used for, how to define them and common ways to work with them. They'll 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 scissors-paper-rock game and the second is a meme generator. Students review a variety of concepts covered in the course, then 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 professional app developers use.

What's Next? Students explore a wide range of app development resources, from the Apple Developer homepage to videos from the Apple Worldwide Developers Conference about the latest frameworks and tools for building apps for all Apple platforms.

Course Outlines (continued)

App Development with Swift

This one-semester course features 45 lessons, each designed to teach a specific skill related to either Swift or app development. Each type of lesson takes a different approach to learning.

- **Swift lessons.** These lessons focus on specific concepts. The labs for each are presented in playgrounds so that students can experiment with code and see the results immediately. Playground files are provided.
- **App development lessons.** Focusing on building specific features for iOS apps, these lessons typically take students step by step through a mini-project. The labs help students apply what they learned to a new scenario.

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 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. In the last unit, they'll examine how to design, prototype and architect an app of their own design.

The Teacher Guide includes tips for extending or adapting lessons, increasing collaboration and supporting students who need additional assistance. It also includes downloadable Keynote presentations for each lesson, solution code for the labs, and a rubric for evaluating student work.

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 the guided project, Light, in which they create a simple flashlight 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 into practice in the guided project, 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 enums. They put this knowledge into practice with the guided project, Personality Quiz, a personalised survey that reveals a fun response to the user.

Unit 4 — Tables and Persistence. Students find out about scroll views and table views, and building complex input screens. They also explore how to save data, share data to other apps and work with images in the user's photo library. They use their new skills in the guided project, 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 any type of information, such as a collection, 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, Restaurant, a customisable menu app that displays the available dishes from a restaurant and allows the user to submit an order. This app uses a web service that allows students to 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.

Additional Information

Download the Get Started with Code resources

- Tynker Coding for Kids
- codeSpark Academy
- *Get Started with Code 1*
- *Get Started with Code 2*

Download the Swift Playgrounds resources

- Learn to Code 1 & 2 iTunes U Course
- *Learn to Code 1 & 2 Teacher Guide*
- *Learn to Code 3 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 the Everyone Can Code program.
- Connect with other educators in the Apple Developer Forums.

About Swift

Swift is the powerful and intuitive programming language created by Apple for building apps. It makes programming easier, more flexible and more fun. Swift is not only great for getting you started with coding, it's also truly 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.

Features are subject to change. Some features may not be available in all regions or all languages.

© 2017 Apple Inc. All rights reserved. Apple, the Apple logo, iPhone, iTunes U, Keynote, Mac, macOS and Xcode are trademarks of Apple Inc., registered in the US and other countries. Swift and Swift Playgrounds are trademarks of Apple Inc. App Store is a service mark of Apple Inc., registered in the US and other countries. Other product and company names mentioned herein may be trademarks of their respective companies. Product specifications are subject to change without notice. This material is provided for information purposes only; Apple assumes no liability related to its use. October 2017