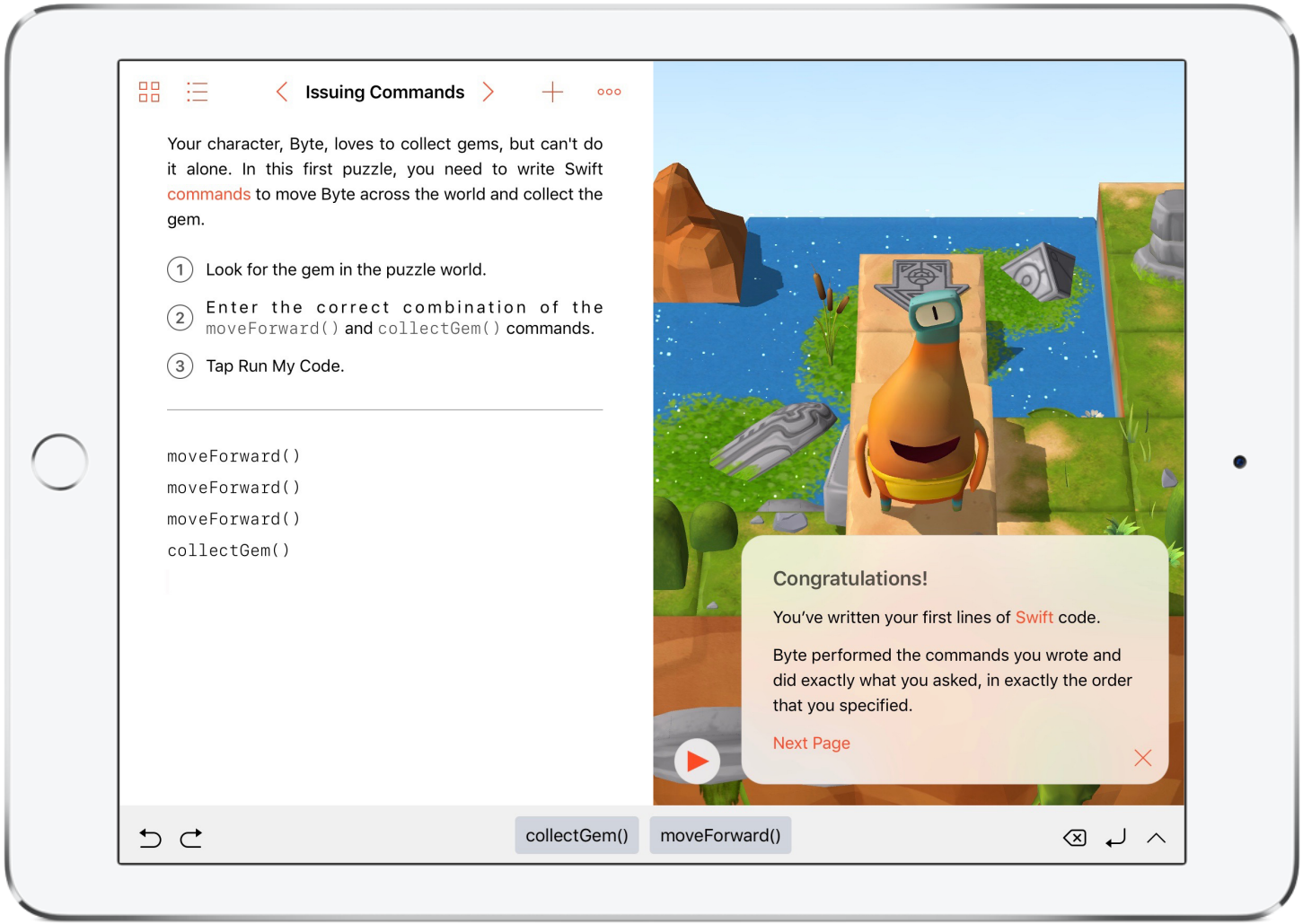
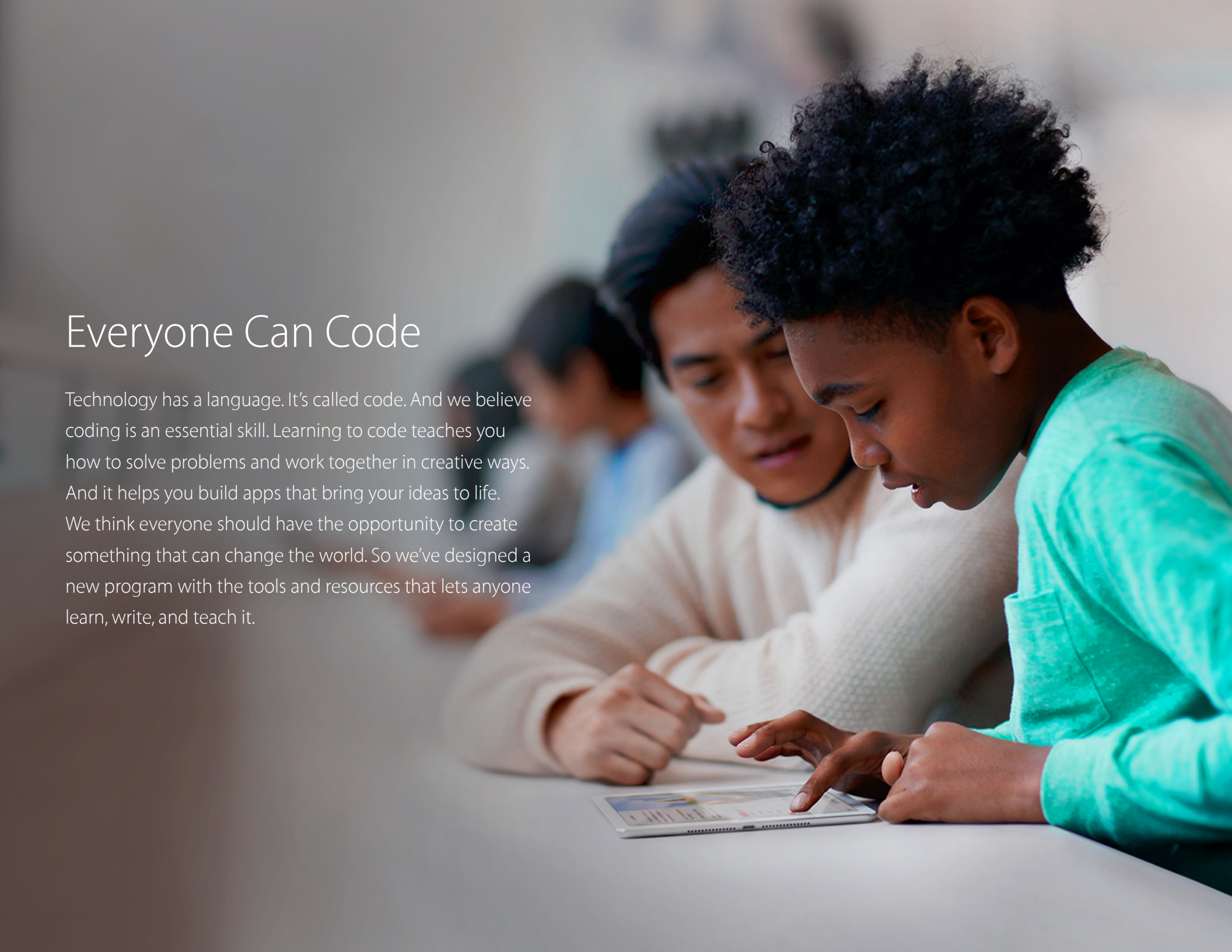
Swift Playgrounds Resource Guide

February 2017



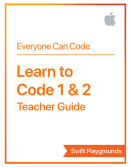
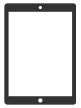
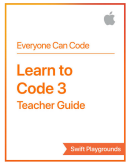
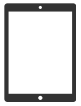
Everyone Can Code

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 lets anyone learn, write, and teach it.



Everyone Can Code Program

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.

Resource	Device	App	Audience	Prerequisites	Overview	Learning materials	Support resources	Number of lesson hours included
			Middle school and up	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	45 hours, including Teacher Guide and Learn to Code 1 & 2 lessons
			Middle school and up	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	20 hours, including Teacher Guide and Learn to Code 3 lessons
			High school and college	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 Xcode project files	• Intro to App Development with Swift: Teacher Guide • Professional learning workshops	90 hours

Overview

Swift Playgrounds is a free iPad app from Apple that makes learning and experimenting with code interactive and fun. Students can solve puzzles to master the basics using Swift—a powerful programming language created by Apple and used by the pros to build many of today's most popular apps.

The app comes with a complete set of Apple-designed lessons called Learn to Code. Using real Swift code, students solve puzzles and meet characters they can control with just a tap. By exploring and solving rich puzzle worlds, students develop coding skills that become the foundation of their programming knowledge.

In Learn to Code 1 & 2, students learn concepts such as commands, debugging, functions, loops, algorithms, and more. The lessons require no previous experience, so they're perfect for first-time coders. Learn to Code 3 helps students expand their coding skills to start thinking more like an app developer.

In the classroom

Learn to Code 1, 2 & 3, along with the lessons in the Teacher Guides, are for use with students in middle school and above. The materials are flexible and usable in any learning environment, and can be used in a stand-alone coding class or as part of an intro to coding program. Lessons are designed for 45- to 60-minute class periods, and some span multiple class periods. The suggested amount of time needed to complete each section in a lesson is included, so if you teach a less-structured class, like an after-school program, you can divide up the lesson.

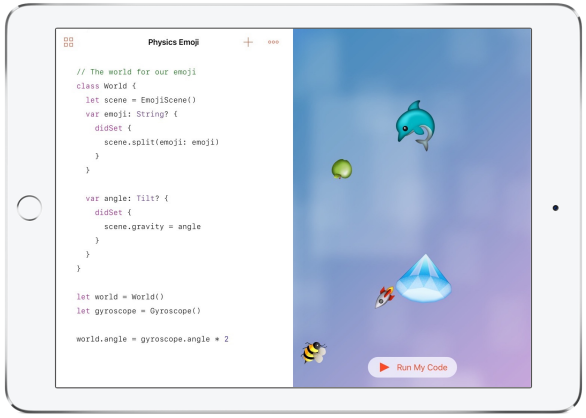
The Teacher Guide provides support that allows teachers with or without coding experience to teach with it. It's recommended that students and teachers have fundamental knowledge of the coding concepts taught in Learn to Code 1 & 2 before moving on to Learn to Code 3.

Swift Playgrounds includes built-in coding lessons and stand-alone challenges.

Key Features

Real Swift, real iOS code. At the heart of Swift Playgrounds is the same Swift programming language that's used to build many of the leading apps in the App Store today. The skills students learn in Swift Playgrounds don't just translate into useful skills elsewhere, they're the exact skills they need to build apps.

Interactive environment. Create code on the left side of the screen and instantly see the results on the right, with just a tap.



Accessibility. Swift Playgrounds was designed with accessibility in mind from day one. It takes advantage of the many powerful accessibility features of iOS, including Switch Control and VoiceOver, and even provides additional voice commentary on the actions of characters as students control them with code.

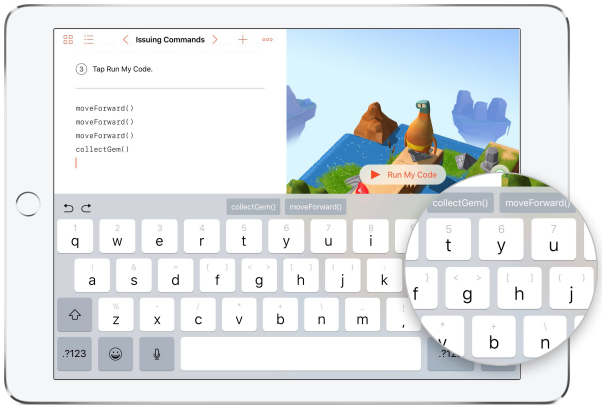
Immersive animations. Each section starts with an immersive animation that relates coding concepts to real life, aiding in student understanding.

Built-in glossary. Definitions help students understand specific terms.

Helpful hints. Students can get help along the way if they get stuck. In many cases, hints change dynamically as they enter code.

Shortcut Bar. QuickType suggestions for code appear at the bottom of the screen that let students enter the code they need by just tapping the Shortcut Bar.

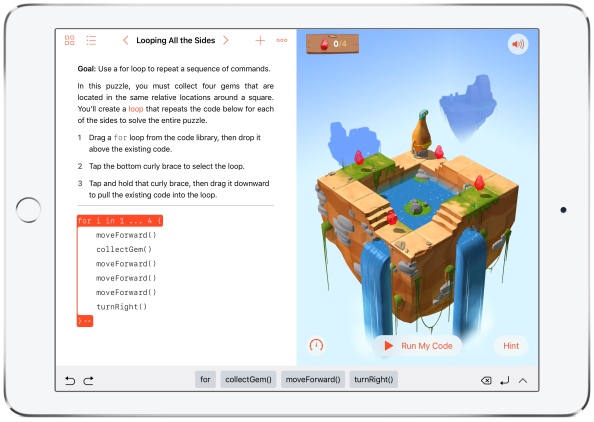
Onscreen keyboard. A keyboard dedicated to Swift provides quick access to the numbers and symbols most commonly used in Swift.



Record and share. Students can record what they do onscreen to demonstrate their work.

Review code. Run code faster or slower, or step through it to highlight the lines of code as they execute, making it easier for students to identify where errors might occur.

Touch to edit. Drag complex structures that wrap other code, such as loops and function definitions, around existing code. Just touch the keyword (such as "for") and the drag controls appear onscreen.



Edit in place. Edit numeric values, colours, and operators quickly and easily using a popover keypad.

Support Resources

Learn to Code 1 & 2: Teacher Guide

Designed for use with students in middle school and up, this Teacher Guide will help any teacher bring Learn to Code 1 & 2 into their classroom. The lessons highlight key coding concepts while demonstrating how coding is a way of thinking that can be applied to other subjects and everyday life. Enhanced activities, review and reflection activities, a grading rubric, and Keynote presentations are included. The lessons represent 45 hours of instruction. Correlation maps in the appendix provide a preliminary alignment of the Learn to Code 1 & 2 lessons to the British Columbia Applied Design, Skills and Technologies grade 6 and 7 standards, as well as to the US Computer Science Teachers Association's Interim Computer Science Standards for Level 2.

Learn to Code 3: Teacher Guide

This guide is geared toward middle school and above and includes 20 hours of lessons. Building on coding skills from Learn to Code 1 & 2, it includes story activities, code review lessons, Keynote presentations, journal prompts, a grading rubric, and more to help teachers bring these lessons into the classroom. A correlations map in the appendix provides a preliminary alignment of the Learn to Code 3 lessons to the U.S. Computer Science Teachers Association's Interim Computer Science Standards for Level 2.

Learn to Code 1 & 2 iTunes U course

This iTunes U course brings the Learn to Code 1 & 2 Teacher Guide to life through video lessons and additional resources. The videos are also a great way for teachers to see how to bring the Teacher Guide lessons to life in a classroom.

Course Outlines

Learn to Code 1

By solving puzzles in a dynamic 3D puzzle world, students will develop a set of coding skills to build up their basic programming vocabulary. Their coding journey begins with simple commands, functions, and loops. From the start, they'll write real Swift code—the same code used by real programmers.

Lesson 0—Getting Started. Students get an introduction to computer science and the goals of the course.

Lesson 1—Think Like a Computer: Commands and Sequences. Students learn about the use of commands and sequences in an everyday situation, then code using commands and sequences.

Lesson 2—Think Like a Detective: Debugging. Students learn about the use of debugging in an everyday situation, then debug with code.

Lesson 3—Think Efficiently: Functions and a Bit of Loops. Students learn about the use of functions and for loops in an everyday situation, then code using functions and for loops.

Review and Reflect. Students review Lessons 1 through 3, review their portfolios, and create a community with peer-to-peer review.

Lesson 4—Thinking Logically: Conditional Code. Students learn about the use of conditional code, Booleans, and logical operators, then code using conditional code, Booleans, and logical operators.

Lesson 5—Think Again and Again: While Loops. Students learn about the use of while loops in an everyday situation, then code using while loops.

Lesson 6—Think the Same Idea: Algorithms. Students learn about the use of algorithms in an everyday situation, then code using algorithms.

Review and Reflect. Students review coding concepts from Lessons 3 through 6, continue reflection on their portfolios, and continue their community experience.

Learn to Code 2

Students will build on their fundamental knowledge of Swift. They'll journey beyond simply solving puzzles and create worlds of their own. They'll learn about variables and types, the coding constructs that allow them to store and access information. These new skills, along with initialization and parameters, will give them even more ways to use code to interact with their characters and the puzzle world, allowing them to change the rules of the world itself.

Lesson 7—Think Like a NewsBot: Variables. Students learn about the use of variables in an everyday situation, then code using variables.

Lesson 8—Think Like an Architect: Types. Students learn about the use of types in an everyday situation, then code using types and initialization.

Lesson 9—Think Specifically: Parameters. Students learn about the use of parameters in an everyday situation, then code using parameters.

Lesson 10—Think Organized: Arrays. Students learn about the use of arrays in an everyday situation, then code using arrays.

Milestone Project. Students build their own worlds using the concepts learned throughout the program, creating a story to go with the world. They reflect on what they've learned using both their portfolios and the community peer-to-peer review.

Course Outlines (continued)

Learn to Code 3

Learn to Code 3 helps students expand the coding skills they learned in previous lessons to start thinking more like an app developer. Learn to Code 1 & 2 is a recommended prerequisite to Learn to Code 3.

Encountering the interstellar space of Blu's universe, students build a set of creative tools as they explore powerful coding concepts that professional developers use. As they learn about graphics and coordinates, they'll be able to place and manipulate images, then combine these techniques with touch events to paint artistic shapes in space.

After learning about touch events, students dive into strings, giving them a way to bring their voice into Blu's silent universe. Finally, they'll explore event handlers as they use real events, such as finger movements or taps, to trigger their code. With event handlers, they'll create animated aliens or turn the universe into a giant musical instrument. By the time they finish, they'll be combining their skills expertly, writing their most advanced code yet!

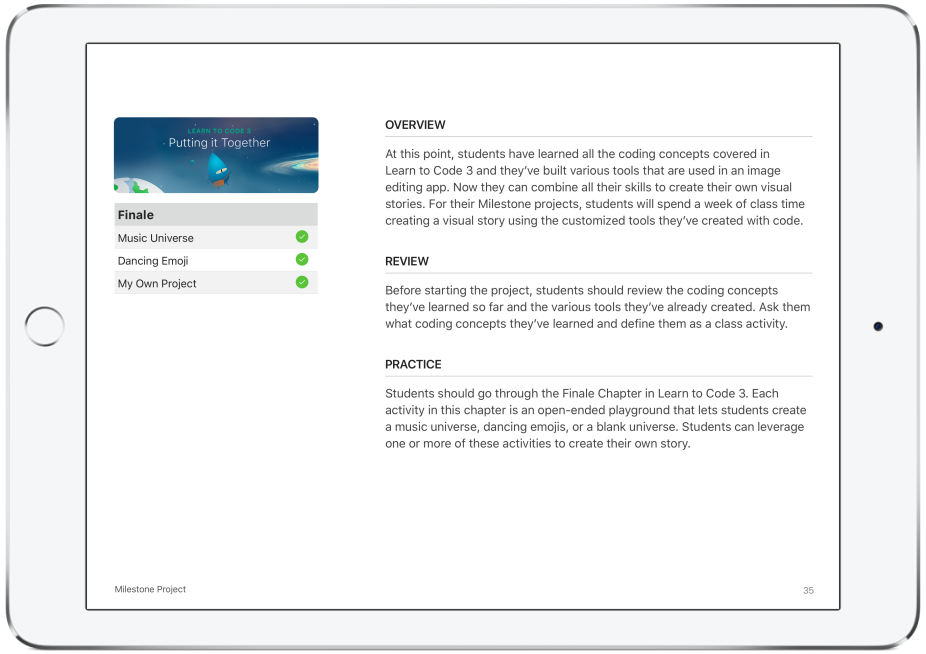
Lesson 1—Introduction to Learn to Code 3: Coordinates. Students learn about coordinates; review algorithms, for loops, and arrays; and then code using a combination of concepts. They also discuss what makes a great visual story.

Lesson 2—Think Like an App Designer: Touch Events. Students review variables, types, and initialization by analyzing their favourite apps, then create and initialize their own image tools in Swift Playgrounds. They also research how images impact visual stories.

Lesson 3—Think Like an Editor: Strings. Students learn about strings in an everyday situation, then create their own text tools in Swift Playgrounds. They also research how text impacts visual stories.

Lesson 4—Think Like an Animator: Event Handlers. Students learn about event handlers by designing their own games. They then create their own action tools in Swift Playgrounds and research how interactivity impacts visual stories.

Milestone Project. Students code their own visual stories in Swift Playgrounds.



Additional Information

Swift Playgrounds requires iOS 10 and works on the following iPad models:

- iPad Pro (9.7-inch)
- iPad Pro (12.9-inch)
- iPad Air 2
- iPad Air
- iPad mini 4
- iPad mini 3
- iPad mini 2

Download the Swift Playgrounds resources

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

Additional resources

- Learn more about Swift Playgrounds.
- Learn more about the Everyone Can Code program.
- Learn more about Swift.
- Connect with other educators in the Apple Developer Forums.

Curriculum Alignment: Learn to Code 1 & 2

Here is the preliminary alignment of the Learn to Code 1 & 2 lessons to the British Columbia Applied Design, Skills and Technologies grade 6 and 7 standards.

British Columbia Ministry of Education Learn to Code 1 & 2 CORRELATION MAP	Applied Design, Skills and Technologies (Grades 6–7)						
	Understand: Big Ideas	Do: Core and Curricular Competencies					Know: Content Learning Standards
	Core Competencies	Curricular Competencies					
Students should be able to:	**Understanding context**	**Defining**	**Ideating**	**Prototyping**	**Testing**	**Making**	**Sharing**
Other Curricular Competencies The following competencies are not a primary focus of the coding resources: • Demonstrate an awareness of precautionary and emergency safety procedures in both physical and digital environments • Identify and evaluate the skills and skill levels needed, individually or as a group, in relation to a specific task, and develop them as needed • Select, and as needed learn about, appropriate tools and technologies to extend their capability to complete a task • Identify the personal, social, and environmental impacts, including unintended negative consequences, of the choices they make about technology use • Identify how the land, natural resources, and culture influence the development and use of tools and technologies	• Empathize with potential users to find issues and uncover needs and potential design opportunities	• Choose a design opportunity • Identify key features or potential users and their requirements • Identify criteria for success and any constraints	• Generate potential ideas and add to others' ideas • Screen ideas against criteria and constraints • Evaluate personal, social, and environmental impacts and ethical considerations • Choose an idea to pursue	• Identify and use sources of information • Develop a plan that identifies key stages and resources • Explore and test a variety of materials for effective use • Construct a first version of the product or a prototype, as appropriate, making changes to tools, materials, and procedures as needed • Record iterations of prototyping	• Test the first version of the product or the prototype • Gather peer and/or user and/or expert feedback and inspiration • Make changes, troubleshoot, and test again	• Identify and use appropriate tools, technologies, and materials for production • Make a plan for production that includes key stages, and carry it out, making changes as needed • Use materials in ways that minimize waste	• Decide on how and with whom to share their product • Demonstrate their product and describe their process, using appropriate terminology and providing reasons for their selected solution and modifications • Evaluate their product against their criteria and explain how it contributes to the individual, family, community, and/or
○	◑	◕	◑	◔	●	◑	◑

KEY ⓘ Information ● 100% coverage ◕ 75% coverage ◑ 50% coverage ◔ 25% coverage ○ Further Notes

Curriculum Alignment: Learn to Code 1 & 2

Here is the preliminary alignment of the Learn to Code 1 & 2 lessons to the British Columbia Applied Design, Skills and Technologies grade 6 and 7 standards.

British Columbia Ministry of Education Learn to Code 1 & 2 CORRELATION MAP	Applied Design, Skills and Technologies (Grades 6–7)			
	Understand: Big Ideas		Do: Core and Curricular Competencies	Know: Content Learning Standards
Students are expected to know the following:	simple algorithms that reflect computational thinking	visual representations of problems and data	evolution of programming languages	visual programming
Overall Alignment	◔	◑	○	○
Commands ⓘ	◔	◑		
Functions ⓘ	◔	●		
For Loops ⓘ	◑	●		
Conditional Code ⓘ	◕	●		
Logical Operators ⓘ	◕	●		
While Loops ⓘ	◕	◑		
Algorithms ⓘ	◕	●		
Variables ⓘ	◕	●		
Types ⓘ	◕	●		
Initialization ⓘ	◕	●		
Parameters ⓘ	◕	●		
World Building ⓘ	◕			
Arrays ⓘ	◕	●		

KEY ⓘ Information ● 100% coverage ◕ 75% coverage ◑ 50% coverage ◔ 25% coverage ○ Further Notes

Curriculum Alignment: Learn to Code 1, 2 & 3

Here is the preliminary alignment of Swift Playgrounds Learn to Code 1, 2 & 3 Teacher Guides with the U.S. Computer Science Teachers Association's (CSTA) Interim Computer Science Standards for Level 2. Once the new standards are finalized, the guide will undergo CSTA's formal crosswalk review. The alignment covers the algorithms and programming concepts within the Computer Science Interim 2016 CSTA K–12 Standards.

CSTA K–12 Computer Science Standards Level 2 for Grades 6–8										
Swift Playgrounds Chapters	**CSTA Standard**	**2-A-2-1** Peer Feedback	**2-A-7-2** Algorithms: Compare	**2-A-7-3** Attribution	**2-A-7-4** Algorithms: Interpret	**2-A-5-5** Computational Artifacts	**2-A-5-6** Develop Programs	**2-A-5-7** Variables	**2-A-4-8** Parameters	**2-A-3-9** Decomposition	**2-A-6-10** Iterative Design
	Commands	●	●							●	●
	Functions	●	●		●		●		●	●	●
	For Loops	●	●		●		●		●	●	●
	Conditional Code	●	●		●		●		●	●	●
	Logical Operators	●	●		●		●		●	●	
	While Loops	●	●		●		●		●	●	●
	Algorithms	●	●		●		●		●	●	●
	Variables	●	●		●		●	●	●	●	●
	Types	●	●		●		●	●	●	●	●
	Initialization	●	●						●	●	
	Parameters	●	●		●		●		●	●	
	World Building	●	●		●		●		●	●	
	Arrays	●	●		●		●	●	●	●	●
	Coordinates	●	●		●	●	●	●	●	●	●
	Touch Events	●	●	●	●	●	●	●	●	●	●
	Strings	●	●	●	●	●	●	●	●	●	●
	Event Handlers	●	●	●	●	●	●	●	●	●	●
	Finale	●	●		●	●	●	●	●	●	●

Key: ● Aligns to standard

Curriculum Alignment: Learn to Code 1 & 2

Throughout the Swift Playgrounds Learn to Code 1 & 2 puzzles and Teacher Guide lessons, students develop an understanding of programming concepts, such as functions, algorithms, variables, types, conditional code, loops, and arrays. Students learn to recognize and define computational problems by abstracting from everyday activities. They develop and use abstractions by decomposing problems into smaller solutions, modelling them as functions and processes, and finally creating their own computational artifacts. Students are encouraged to work with algorithms, predicting their outcomes, exchanging solutions and testing, and refining their work by soliciting peer feedback to detect bugs and improve outcomes.

CSTA K–12 Computer Science Standards Level 2 for Grades 6–8										
Swift Playgrounds Chapters	**CSTA Standard**	**2-A-2-1** Peer Feedback	**2-A-7-2** Algorithms: Compare	**2-A-7-3** Attribution	**2-A-7-4** Algorithms: Interpret	**2-A-5-5** Computational Artifacts	**2-A-5-6** Develop Programs	**2-A-5-7** Variables	**2-A-4-8** Parameters	**2-A-3-9** Decomposition	**2-A-6-10** Iterative Design
	Commands	●	●							●	●
	Functions	●	●		●		●		●	●	●
	For Loops	●	●		●		●		●	●	●
	Conditional Code	●	●		●		●		●	●	●
	Logical Operators	●	●		●		●		●	●	
	While Loops	●	●		●		●		●	●	●
	Algorithms	●	●		●		●		●	●	●
	Variables	●	●		●		●	●	●	●	●
	Types	●	●		●		●	●	●	●	●
	Initialization	●	●						●	●	
	Parameters	●	●		●		●		●	●	
	World Building	●	●		●		●		●	●	
	Arrays	●	●		●		●	●	●	●	●

Key: ● Aligns to standard

Curriculum Alignment: Learn to Code 3

Throughout the Swift Playgrounds Learn to Code 3 puzzles and Teacher Guide lessons, students are introduced to coordinates, touch events, strings, and event handlers. They begin to refine collaboration skills by working on activities in groups and by exchanging and analyzing one another's code. They continue to develop their communication skills by journaling and presenting their progress using video, diagrams, and charts.

Students continue to develop and refine programs by soliciting peer feedback, comparing algorithms and solutions, and exploring issues of speed, clarity, and size. They move on to analyzing each other's code for clarity, developing a "Genius Bar" to leverage peer feedback and input.

They begin to create their own visual stories using iterative design principles. While working through the coordinates and touch events chapters, students are encouraged to use their full coding toolkit of algorithms, for loops, conditional code, variables, types, and arrays. As they create images using coordinates, learn about touch events, strings, and event handlers, they begin to understand how interactivity, text, and audio contribute to creating powerful visual stories.

CSTA K–12 Computer Science Standards Level 2 for Grades 6–8										
	CSTA Standard	**2-A-2-1** Peer Feedback	**2-A-7-2** Algorithms: Compare	**2-A-7-3** Attribution	**2-A-7-4** Algorithms: Interpret	**2-A-5-5** Computational Artifacts	**2-A-5-6** Develop Programs	**2-A-5-7** Variables	**2-A-4-8** Parameters	**2-A-3-9** Decomposition	**2-A-6-10** Iterative Design
Swift Playgrounds Chapters	**Coordinates**	●	●		●	●	●	●	●	●	●
	Touch Events	●	●	●	●	●	●	●	●	●	●
	Strings	●	●	●	●	●	●	●	●	●	●
	Event Handlers	●	●	●	●	●	●	●	●	●	●
	Finale	●	●		●	●	●	●	●	●	●

Key: ● Aligns to standard

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