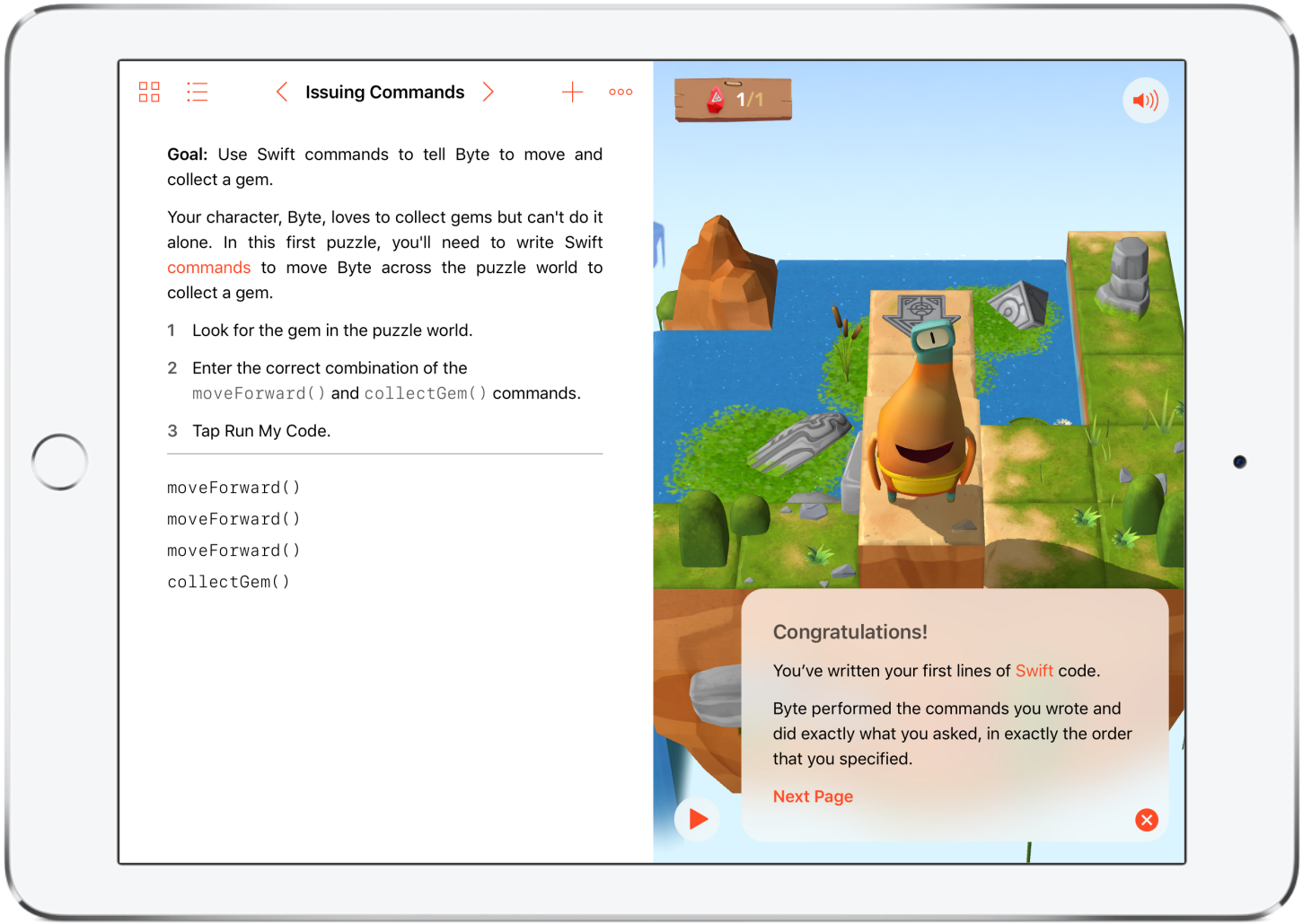
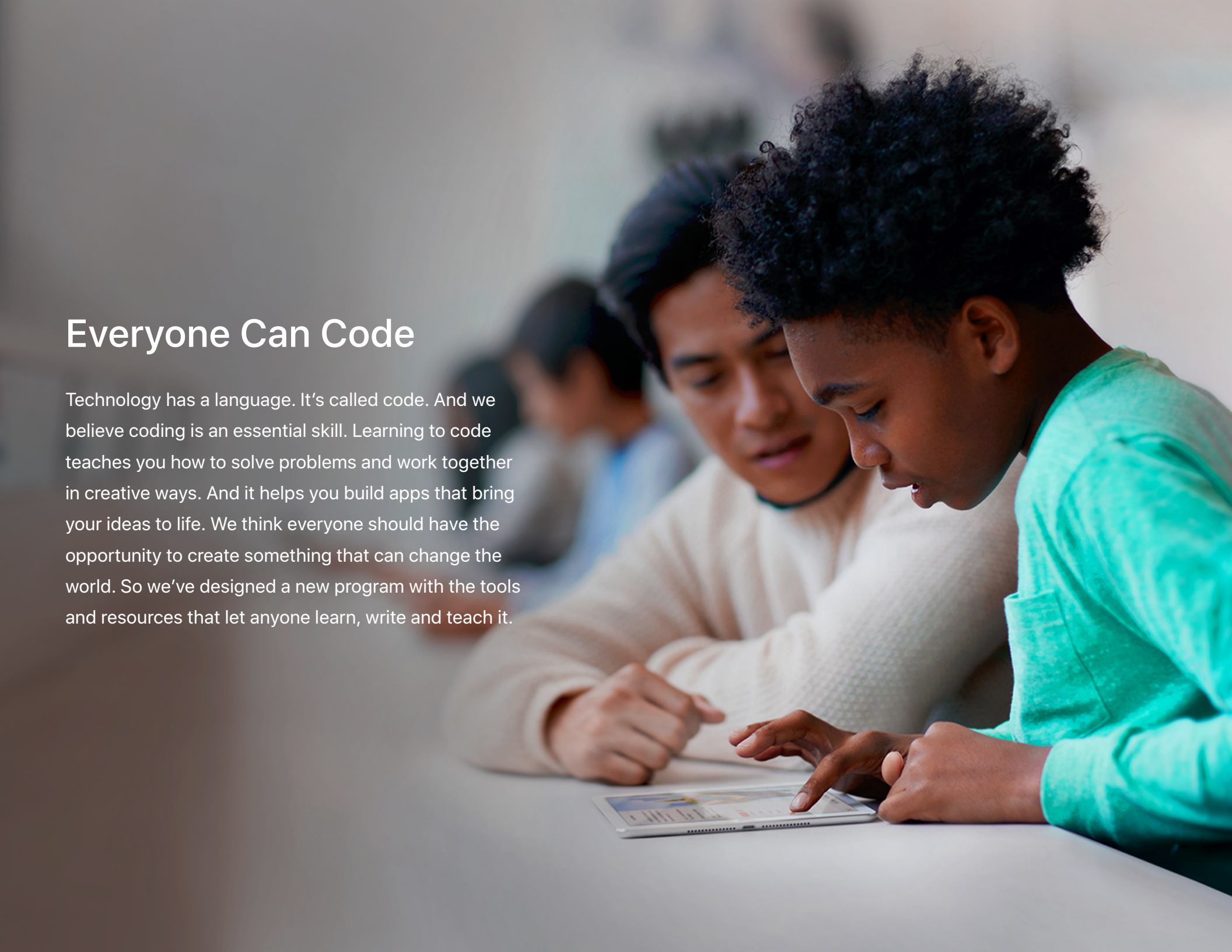
Swift Playgrounds Curriculum Guide

October 2017



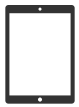
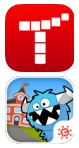
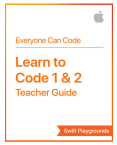
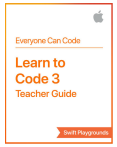
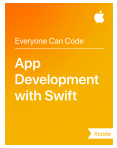
Everyone Can Code

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

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. Additional challenges and connected device playgrounds are also included to let students apply what they've learned to new contexts.

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. An optional app design section in the Teacher Guide helps teachers lead students through an app design process.

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

In the classroom

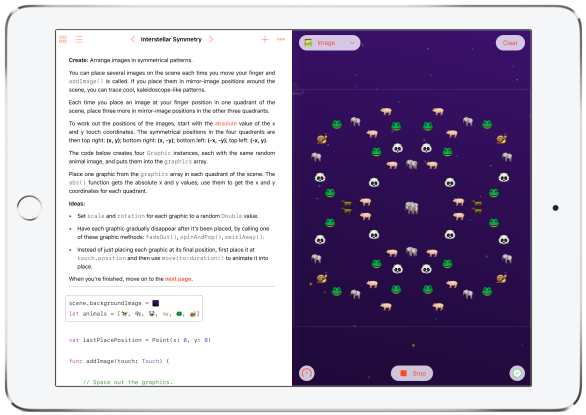
Learn to Code 1, 2 & 3, along with the lessons in the Teacher Guides, are for use with older primary school and junior high school students. The materials are flexible and usable in any learning environment, and can be used in a stand-alone coding class or as part of an intro to coding program. Lessons are designed for 45- to 60-minute classes, and some span multiple 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 Guides provide support that allows teachers with or without coding experience to teach with them. 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.

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.

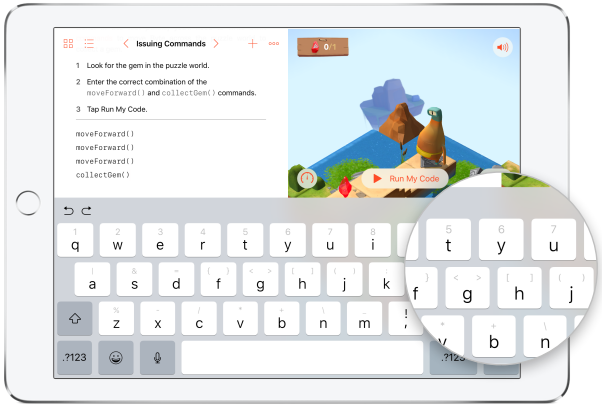
Immersive animations. Each section starts with an immersive animation that relates coding concepts to real life, aiding student understanding.

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.

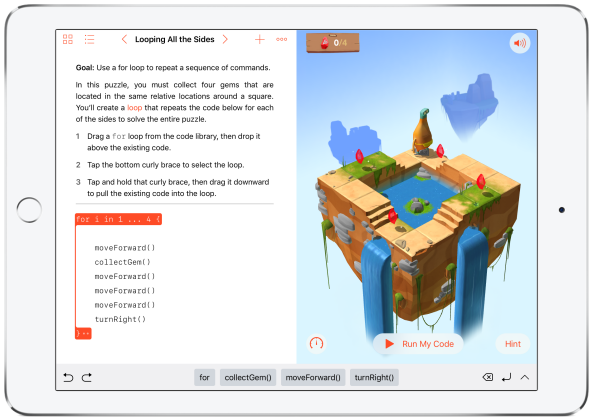
Onscreen keyboard. A keyboard dedicated to Swift provides quick access to the numbers and symbols most commonly used in Swift.



Record and share. Students can record what they do on screen 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 on screen.



Edit in place. Edit numeric values, colours and operators quickly and easily using a pop-over keypad.

Support Resources

Learn to Code 1 & 2 Teacher Guide

Designed for use with older primary school and junior high school students, 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 guide represents 40 to 45 hours of core coding lessons, with up to 45 hours of supplemental activities that help students apply what they've learned and start designing their very own app. Curriculum correlations are included, showing alignment with various national and international curriculum standards for computer science.

Learn to Code 3 Teacher Guide

The guide is geared toward older primary school and junior high school students. It includes 20 hours of core coding lessons, with up to 25 hours of supplemental activities that help students apply what they've learned and start designing their very own app. 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. Curriculum correlations are included, showing alignment with various national and international curriculum standards for computer science.

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 they can bring the Teacher Guide lessons to life in a classroom.

Apple Teacher Program: Earn Swift Playgrounds badges

The Apple Teacher Program is a free professional learning program designed to support and celebrate teachers. It offers self-paced learning materials, tips, inspiration and news. Apple Teachers can visit the Apple Teacher Learning Center to complete quizzes on learning and teaching with Swift Playgrounds, and earn four new badges. They'll then receive an updated Apple Teacher logo featuring Swift Playgrounds to share their accomplishment.

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 using commands and sequences in an everyday situation, then code using commands and sequences.

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

Lesson 3 — Think Efficiently: Functions and a Bit of Loops. Students learn about using functions and for-loops in an everyday situation, then how to 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 using conditional code, Booleans and logical operators, then how to code using conditional code, Booleans and logical operators.

Lesson 5—Think Again and Again: While Loops. Students learn about using while-loops in an everyday situation, then how to code using while-loops.

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

Review and Reflect. Students review coding concepts from lessons 3 through 6, continue reflecting 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. And they'll learn about variables and types, the coding constructs that allow them to store and access information. These new skills, along with initialisation 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 using variables in an everyday situation, then how to code using variables.

Lesson 8 — Think Like an Architect: Types. Students learn about using types in an everyday situation, then how to code using types and initialisation.

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

Lesson 10 — Think Organised: Arrays. Students learn about using arrays in an everyday situation, then how to 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. Then they reflect on what they've learned using their portfolios and the community peer-to-peer review.

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

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 for 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.

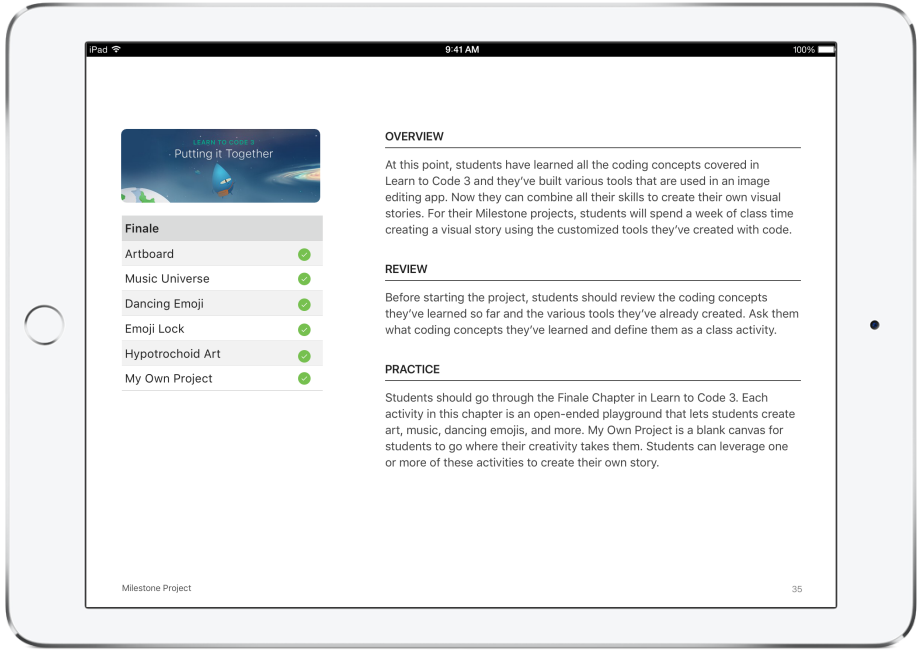
Lesson 2 — Think Like an App Designer: Touch Events. Students review variables, types and initialisation by analysing their favourite apps, then create and initialise their own image tools in Swift Playgrounds. They also research how images influence visual stories.

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 influences 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.

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



Additional Information

Swift Playgrounds requires iOS 10 or later and works on:

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

Download the Get Started with Code resources

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

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 Correlations: Learn to Code 1, 2 & 3

The Learn to Code 1 & 2 and Swift Playgrounds exercises are targeted at older primary school and junior high school students, and align with the Processes and Production Skills strand of the Australian Curriculum: Digital Technologies for years 5–10. Teachers can follow a structured pathway through the Teacher Guides while mapping to the corresponding Digital Technologies content description.

Alignment between Learn to Code 1 & 2 and Australian Curriculum: Digital Technologies — Years 5–10 Processes and Production Skills Strand			
		Content description	
Years 5–6	**ACTDIP016**	Acquire, store and validate different types of data, and use a range of software to interpret and visualise data and create information	●
	ACTDIP017	Define problems in terms of data and functional requirements, drawing on previously solved problems	●
	ACTDIP018	Design a user interface for a digital system	●
	ACTDIP019	Design, modify and follow simple algorithms involving sequences of steps, branching and iteration (repetition)	●
	ACTDIP020	Implement digital solutions as simple visual programs involving branching, iteration (repetition) and user input	●
	ACTDIP021	Explain how student solutions and existing information systems are sustainable and meet current and future local community needs	●
	ACTDIP022	Plan, create and communicate ideas and information, including collaboratively applying agreed ethical, social and technical protocols online	
Years 7–8	**ACTDIP026**	Analyse and visualise data using a range of software to create information, and use structure data to model objects or events	●
	ACTDIP027	Define and decompose real-world problems, taking into account functional requirements and economic, environmental, social, technical and usability constraints	●
	ACTDIP028	Design the user experience of a digital system, generating, evaluating and communicating alternative designs	●
	ACTDIP029	Design algorithms represented diagrammatically and in English, then trace algorithms to predict output for a given input and to identify errors	●
	ACTDIP030	Implement and modify programs with user interfaces that involve branching, iteration and functions in a general-purpose programming language	●
	ACTDIP031	Evaluate how student solutions and existing information systems meet needs, are innovative, and take into account future risks and sustainability	●
	ACTDIP032	Plan and manage projects that create and communicate ideas and information collaboratively online, taking into account safety and social contexts	
Years 9–10	**ACTDIP037**	Analyse and visualise data to create information and address complex problems, then model processes, entities and their relationships using structured data	
	ACTDIP038	Define and decompose real-world problems, taking into account functional and non-functional requirements and interviewing stakeholders to identify needs	●
	ACTDIP039	Design the user experience of a digital system by evaluating alternative designs against criteria including functionality, accessibility, usability and aesthetics	●
	ACTDIP040	Design algorithms represented diagrammatically and in structured English, and validate algorithms and programs using tracing and test cases	●
	ACTDIP041	Implement modular programs, and apply selected algorithms and data structures, including using an object-oriented programming language	●
	ACTDIP042	Critically evaluate how students' solutions and existing information systems and policies consider future risks and sustainability, and provide opportunities for innovation and enterprise	
	ACTDIP043	Create interactive solutions for sharing ideas and information online, taking into account safety, social contexts and legal responsibilities	
	ACTDIP044	Plan and manage projects using an iterative and collaborative approach, identifying risks and considering safety and sustainability	

Key: ● Aligns with curriculum