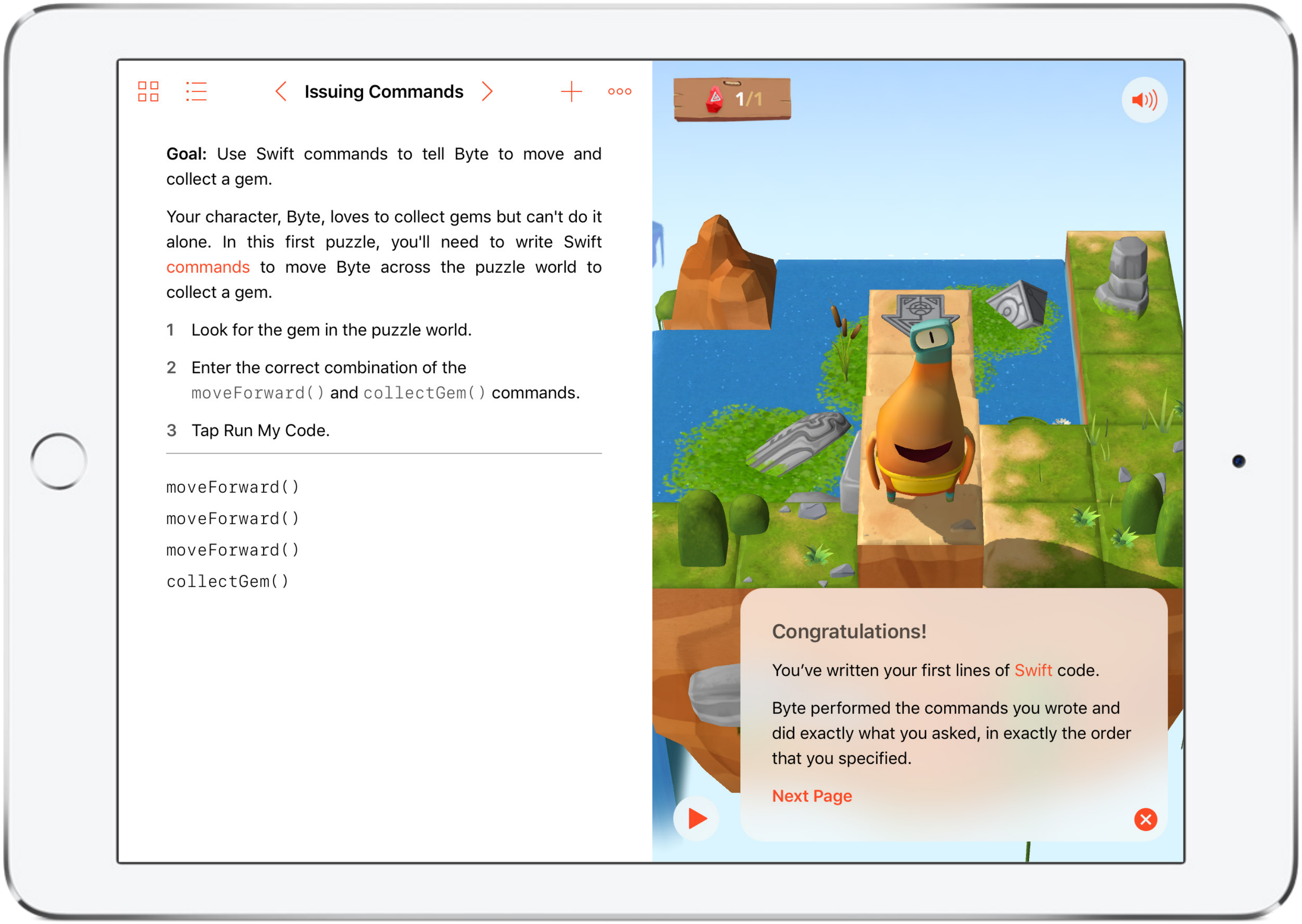
Swift Playgrounds Curriculum Guide

September 2017



Goal: Use Swift commands to tell Byte to move and collect a gem.

Your character, Byte, loves to collect gems but can't do it alone. In this first puzzle, you'll need to write Swift commands to move Byte across the puzzle world to collect a gem.

1 Look for the gem in the puzzle world.

2 Enter the correct combination of the `moveForward()` and `collectGem()` commands.

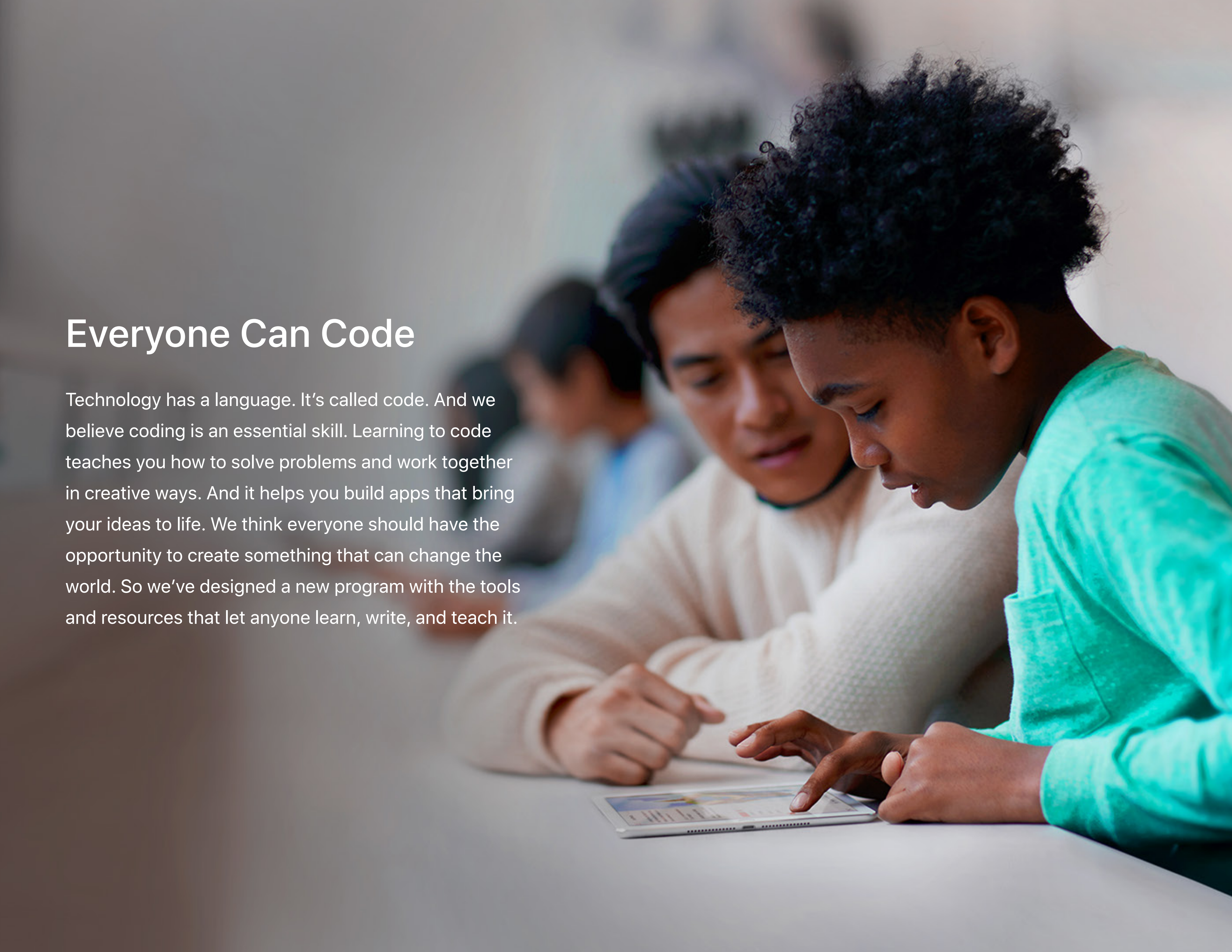
3 Tap Run My Code.

```
moveForward()
moveForward()
moveForward()
collectGem()
```

Congratulations!

You've written your first lines of Swift code.

Byte performed the commands you wrote and did exactly what you asked, in exactly the order that you specified.

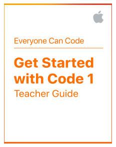
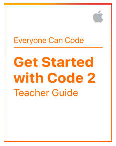
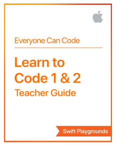
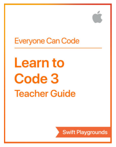


Everyone Can Code

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
 Everyone Can Code Get Started with Code 1 Teacher Guide		Kindergarten through grade 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
 Everyone Can Code Get Started with Code 2 Teacher Guide		Grades 3 through 5		None	Explore fundamental coding concepts and practice 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
 Everyone Can Code Learn to Code 1 & 2 Teacher Guide		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 • Apple Teacher Learning Center Swift Playgrounds badges	Up to 85 hours, including Teacher Guide and Learn to Code 1 & 2 lessons
 Everyone Can Code Learn to Code 3 Teacher Guide		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	Up to 45 hours, including Teacher Guide and Learn to Code 3 lessons
Everyone Can Code Intro to App Development with Swift		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 project files	• Intro to App Development with Swift: Teacher Guide • MobileMakersEdu professional learning workshops	90 hours
Everyone Can Code App Development with Swift		High school and college		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 • MobileMakersEdu professional learning workshops	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.

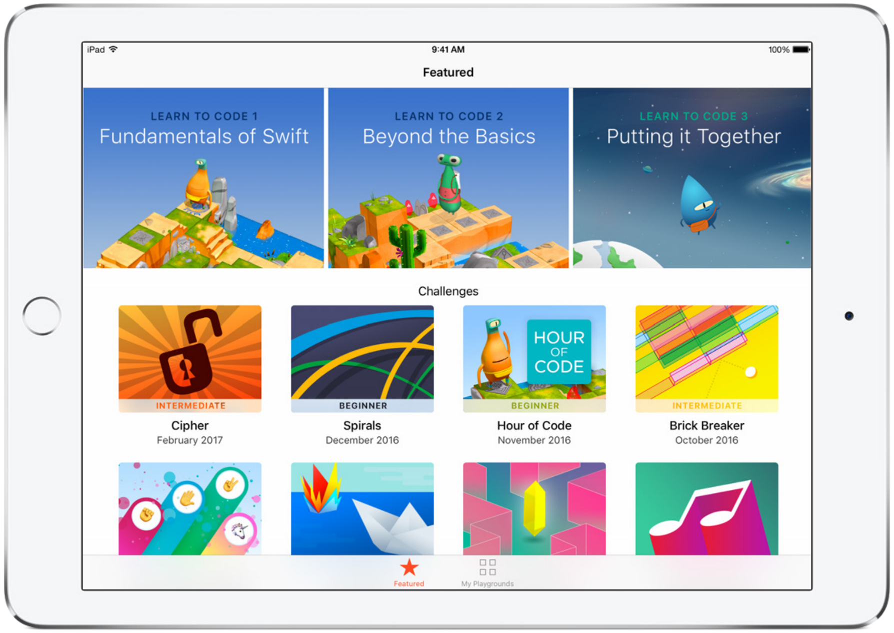
The app comes with a complete set of Apple-designed lessons called Learn to Code. Using real Swift code, students solve puzzles and meet characters they can control with just a tap. By exploring and solving rich puzzle worlds, students develop coding skills that become the foundation of their programming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

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 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.

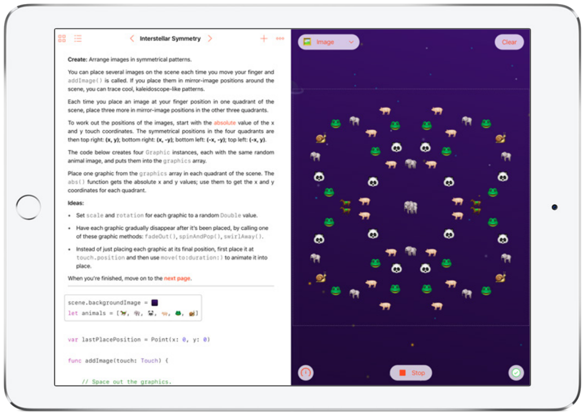


Swift Playgrounds includes built-in coding lessons and stand-alone challenges.

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.



Create: Arrange images in symmetrical patterns.

You can place several images on the scene each time you move your finger and addImage() is called. If you place them in mirror-image positions around the scene, you can trace cool, kaleidoscope-like patterns.

Each time you place an image at your finger position in one quadrant of the scene, place three more in mirror-image positions in the other three quadrants.

To work out the positions of the images, start with the absolute value of the x and y touch coordinates. The symmetrical positions in the four quadrants are then top right: (x, y); bottom right: (x, -y); bottom left: (-x, -y); top left: (-x, y).

The code below creates four Graphic instances, each with the same random animal image, and puts them into the graphics array.

Place one graphic from the graphics array in each quadrant of the scene. The abs() function gets the absolute x and y values; use them to get the x and y coordinates for each quadrant.

Ideas:

- Set scale and rotation for each graphic to a random Double value.
- Have each graphic gradually disappear after it's been placed, by calling one of these graphic methods: fadeOut(), spinAndPop(), swirlAway().
- Instead of just placing each graphic at its final position, first place it at touch.position and then use move(to:duration:) to animate it into place.

When you're finished, move on to the next page.

```
scene.backgroundImage = ■
let animals = [🦋, 🐘, 🐼, 🐖, 🐸, 🐌]

var lastPlacePosition = Point(x: 0, y: 0)

func addImage(touch: Touch) {

    // Space out the graphics.
```

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.

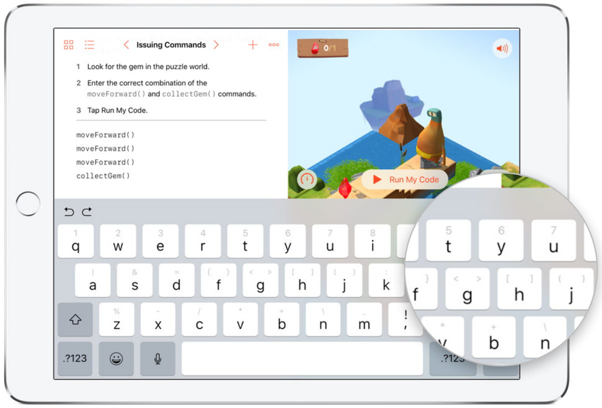
Immersive animations. Each section starts with an immersive animation that relates coding concepts to real life, aiding in student understanding.

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.



1 Look for the gem in the puzzle world.

2 Enter the correct combination of the moveForward() and collectGem() commands.

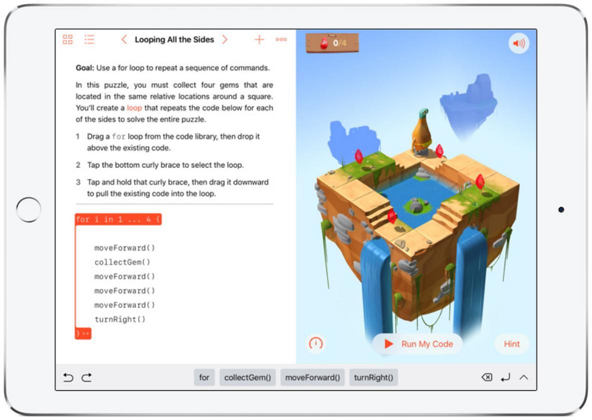
3 Tap Run My Code.

```
moveForward()
moveForward()
moveForward()
collectGem()
```

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.



Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1 Drag a for loop from the code library, then drop it above the existing code.

2 Tap the bottom curly brace to select the loop.

3 Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {

    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
} ··
```

Edit in place. Edit numeric values, colors, 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 guide is geared toward middle school and above. It 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 with alignments to various national and international curriculum standards for computer science.

Learn to Code 3: Teacher Guide

The guide is geared toward middle school and above. 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 with alignments to 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 to 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.

FUNCTION

- A collection of commands grouped together and given a name. The set of commands can then be run with just the name of the function whenever that set is needed.

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.

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

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.

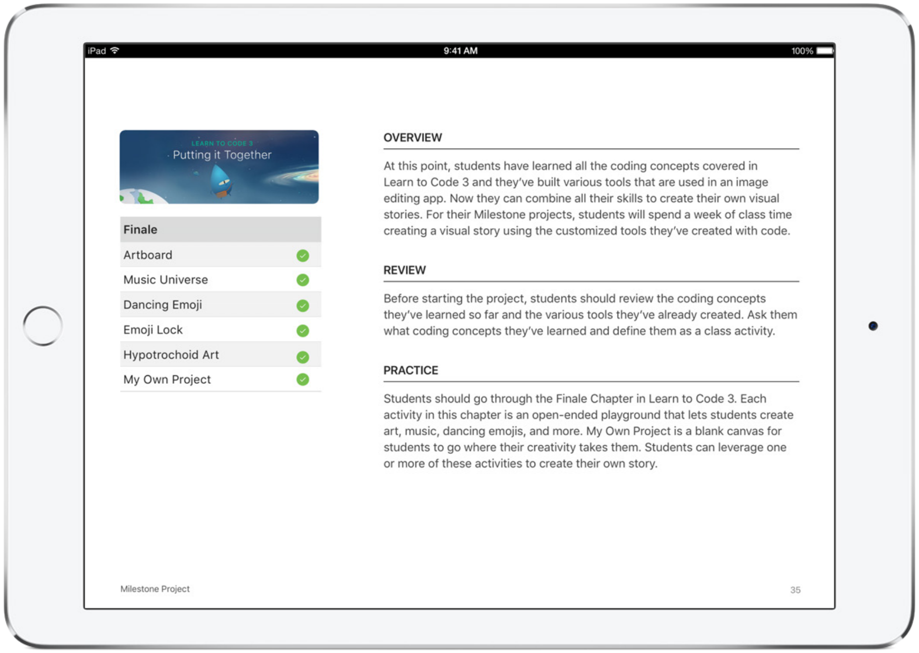
Lesson 2—Think Like an App Designer: Touch Events. Students review variables, types, and initialization by analyzing their favorite apps, then create and initialize their own image tools in Swift Playgrounds. They also research how images impact visual stories.

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.

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



Finale
- Artboard ✓
- Music Universe ✓
- Dancing Emoji ✓
- Emoji Lock ✓
- Hypotrochoid Art ✓
- My Own Project ✓

OVERVIEW

At this point, students have learned all the coding concepts covered in Learn to Code 3 and they've built various tools that are used in an image editing app. Now they can combine all their skills to create their own visual stories. For their Milestone projects, students will spend a week of class time creating a visual story using the customized tools they've created with code.

REVIEW

Before starting the project, students should review the coding concepts they've learned so far and the various tools they've already created. Ask them what coding concepts they've learned and define them as a class activity.

PRACTICE

Students should go through the Finale Chapter in Learn to Code 3. Each activity in this chapter is an open-ended playground that lets students create art, music, dancing emojis, and more. My Own Project is a blank canvas for students to go where their creativity takes them. Students can leverage one or more of these activities to create their own story.

Milestone Project 35

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
- 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
- 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 Alignment: Learn to Code 1, 2 & 3

Here's the preliminary alignment of Swift Playgrounds Learn to Code 1, 2 & 3 Teacher Guides with the Interim 2016 Computer Science Teachers Association (CSTA) K–12 Computer Science Standards for Level 2. 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

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	●	●	●	●	●	●	●	●	●	●
Event Handlers	●	●	●	●	●	●	●	●	●	●
Finale	●	●		●	●	●	●	●	●	●

Key: ● Aligns to standard

Curriculum Alignment: Learn to Code 1 & 2

Here's the preliminary alignment of Swift Playgrounds Learn to Code 1 & 2: Teacher Guide with the Interim 2016 Computer Science Teachers Association (CSTA) K–12 Computer Science Standards for Level 2. 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

	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	**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

Here's the preliminary alignment of Swift Playgrounds Learn to Code 3: Teacher Guide with the Interim 2016 Computer Science Teachers Association (CSTA) K–12 Computer Science Standards for Level 2. 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

	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	●	●	●	●	●	●	●	●	●	●
	Event Handlers	●	●	●	●	●	●	●	●	●	●
	Finale	●	●		●	●	●	●	●	●	●

Key: ● Aligns to standard

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