

Welcome to the Swift Coding Club!

Learning to code teaches you how to solve problems and work with others in creative ways. And it helps you bring your ideas to life.

Swift Coding Clubs are a fun way to learn to code and design apps. Activities built around Swift, Apple's coding language, help you collaborate as you learn to code, prototype apps, and think about how code can make a difference in the world around you.

You don't have to be a teacher or a coding expert to run a Swift Coding Club. The materials are self-paced, so you can even learn alongside your club members. And you can all celebrate your club's ideas and designs with an app showcase event for your community.

This guide is arranged in three sections:



Get Started

Everything you need to launch a Swift Coding Club



Learn and Apply

Modules and activities for club sessions



Celebrate

Helpful resources to plan and host a community event

Coding Resources

Swift Coding Clubs are built around a variety of resources for teaching code. Apple takes coders from learning the basics to building real apps.



Everyone Can Code | Ages 5+

Use Swift code to learn coding fundamentals with Swift Playgrounds on iPad or Mac. [Learn more >](#)



Develop in Swift | Ages 14+

Learn to develop apps in Xcode on Mac. [Learn more >](#)



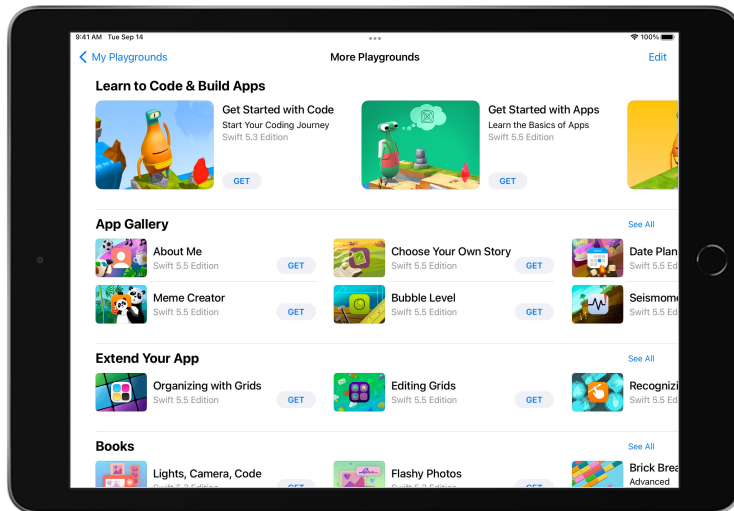
Get Started

1. Explore Everyone Can Code resources

Everyone Can Code introduces learners to the world of coding through interactive puzzles, playful characters, and engaging activities. Before you start designing your club experience, it's helpful to explore the Everyone Can Code resources below.

Swift Playgrounds is a free app that makes learning Swift interactive and fun. It includes a built-in library of lessons and a suite of app projects.

Everyone Can Code Puzzles include activities to introduce coding concepts, connect them to everyday life, and then apply them by solving puzzles in Swift Playgrounds.



[Download and explore Swift Playgrounds >](#)



[Download the Everyone Can Code curriculum >](#)



2. Check your tech

To complete the activities in this guide, you'll need the following:

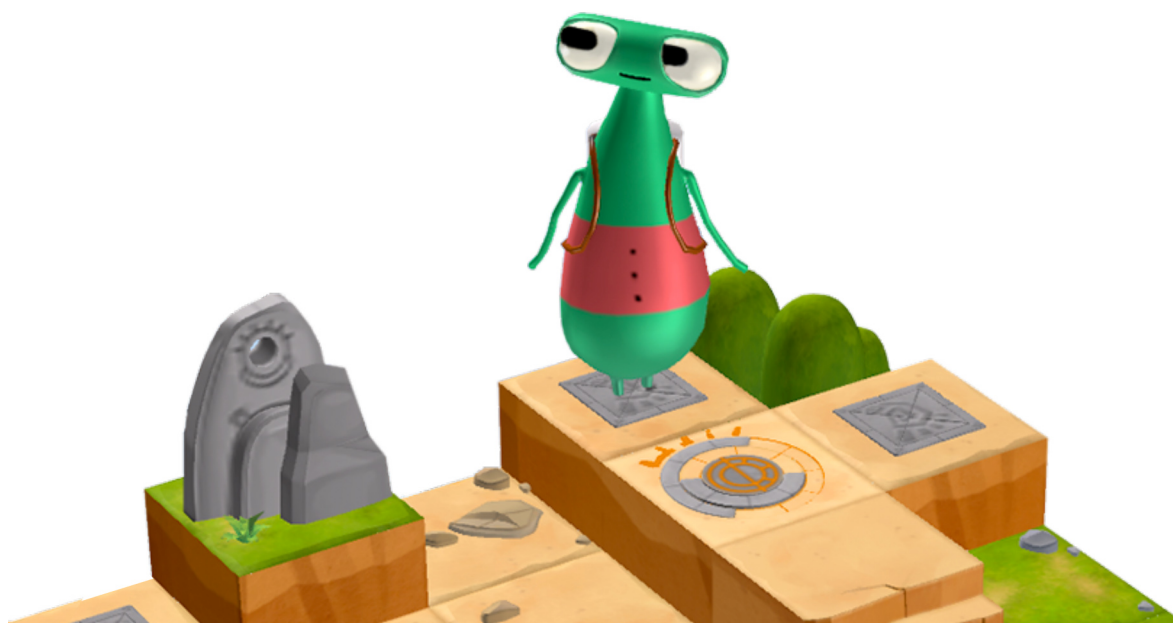
- **iPad or Mac.** Use devices with iPadOS 14.7 or later or macOS 11.5 or later. It's best if each person has their own device, but they can also share and code together.
- **Swift Playgrounds 3.4.1 or later.**
[Download Swift Playgrounds for iPad >](#)
[Download Swift Playgrounds for Mac >](#)
- **Everyone Can Code guides.**
[Download Everyone Can Code Puzzles >](#)

Visit [Apple Support](#) to get help with Apple products.

3. Make a plan

Here are some things to consider:

- Who are your club members? What are their interests? Do they have experience with coding or are they brand-new?
- How often will your club meet? If you're planning a summer camp, how many hours of coding activities will you have?
- What technology is available for the club?
- What are the goals of your club?





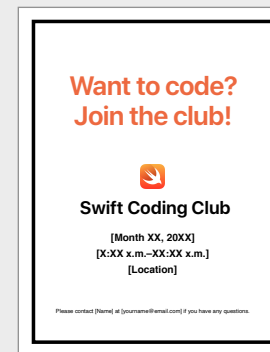
4. Spread the word

Let people know about your Swift Coding Club. Here are some ideas and resources to attract new members to your club:

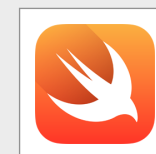
- **Announce your club.** Use email, social media, the web, flyers, or word of mouth to let your community know about your club.
- **Host an informational meeting.** Ask potential club members about their interests and the types of projects they'd want to create. Talk about ideas for holding community events and how club members can get involved. You can also share a short video about the club online.

These items can help you promote and personalize your Swift Coding Club:

- **Posters.** [Download this free template](#), then personalize it to create your own poster. Print and display it, or make a digital poster to share online. Be sure to include details for when and where the club will meet and how to join.
- **Stickers and T-shirts.** Use these [Swift Coding Club stickers](#) to help promote your club. T-shirts are a great way to recognize members who participate in app showcase events. Download the [Swift Coding Club T-shirt template](#) to make shirts for your members.



Swift Coding Club poster



Swift Coding Club sticker



Swift Coding Club T-shirt

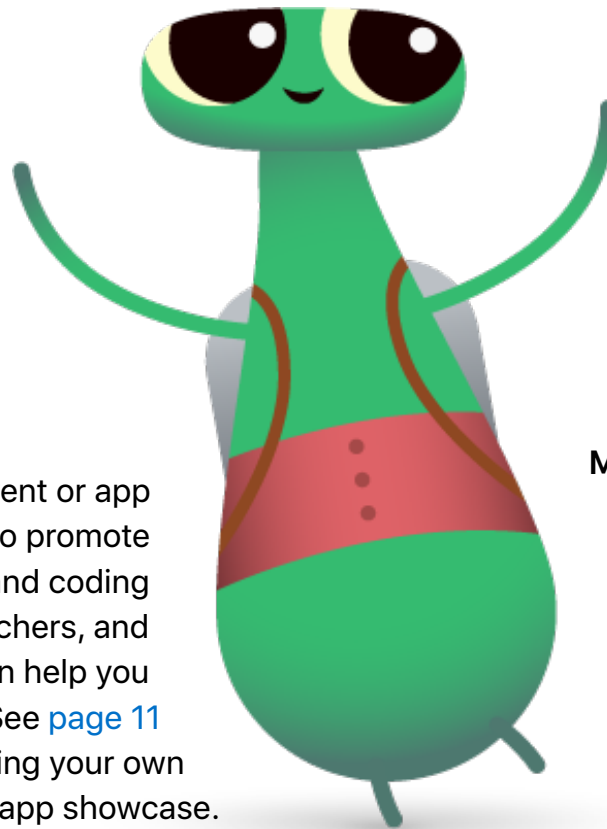
Tips for Club Leaders



Build a leadership team. Having a group of members who help with leading the club can make it much easier and more fun. Which club members have leadership potential? Think about adding officers to your club for events, coding, app design, and more.

Learn together. Club leaders don't have to know everything. Help your members work on their own research and problem-solving skills, and encourage them to help others.

Show off. A community event or app showcase is a great way to promote your club, design ideas, and coding skills to friends, families, teachers, and the community. It might even help you recruit more members. See [page 11](#) to get tips for holding your own community event or app showcase.



Share ideas. Some members will be interested in making games. Others might want to create apps to help people, learn Swift, or control robots. Think about ways for members to work together on projects they care about.

Mix it up. Sometimes members who are more advanced can leave others behind. See if those members can partner up with beginners for pair programming. Teaching someone else is a great way to learn!



Learn and Apply

1. Explore Swift Playgrounds

The club materials are built around Swift Playgrounds, which includes a built-in library of lessons and a suite of app projects. Start by familiarizing yourself with the content in Swift Playgrounds and features of the app.



Swift Playgrounds Features



Snippets Library

To minimize typing, tap in the toolbar to access the Snippets Library and quickly drag commonly used pieces of code.

Tools

Use this menu to reset the page, take a picture, create a PDF, or record a movie.

Pages menu

Tap the page heading to see all playground pages.

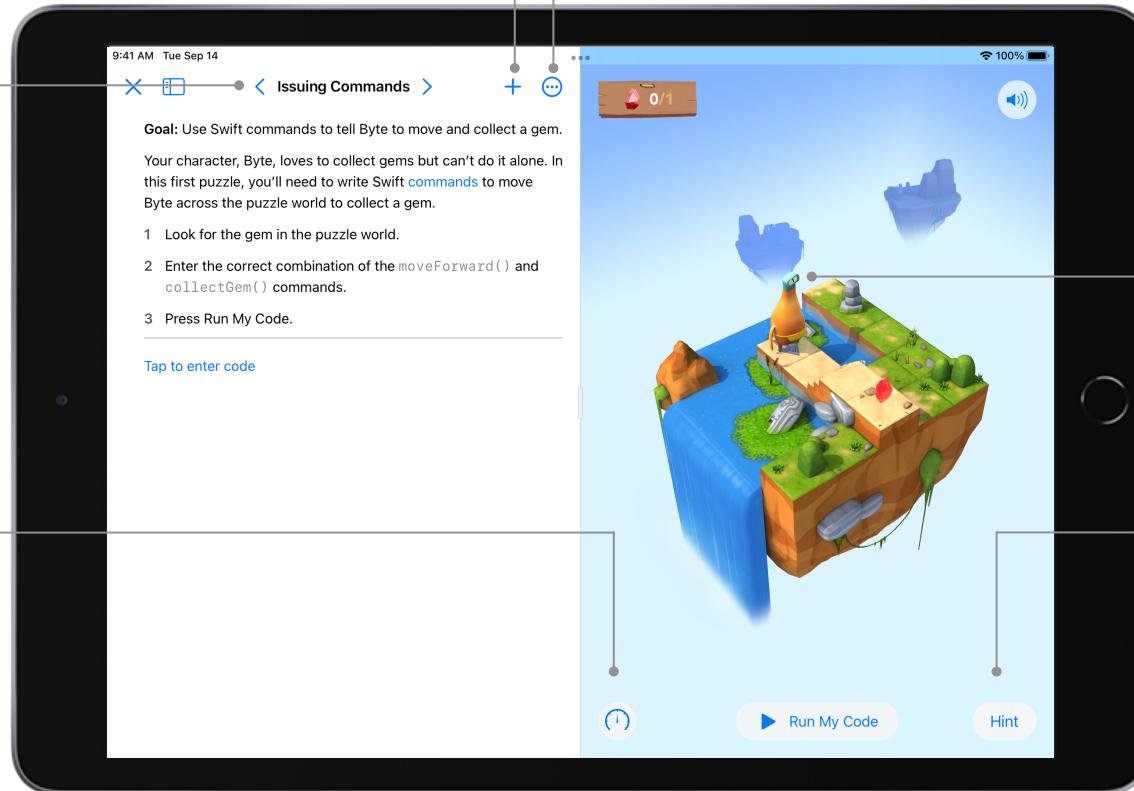
Tap a page or use the arrows to navigate between pages.

Control the speed

Speed up or slow down the code.

Highlight code as it runs

Use Step Through My Code to highlight each line of code as it runs to better understand what the code is doing.



Choose a character

Personalize your experience by tapping the character to choose a different one.

Hint

This feature provides helpful suggestions. And though it also will eventually reveal a puzzle's solution, you can't simply cut and paste the solution. To move on, you still have to complete the steps and write the code yourself.

Tips for Learning with Swift Playgrounds



Explore the puzzles first. Encourage club members to zoom and rotate Byte's world in the live view so they can take a good look at what they need to accomplish. They can also view it full screen by touching and holding the partition between the two windows, then dragging to the left.

Break down the puzzles. The puzzles get tricky. Club members can divide a puzzle into parts to help them think through all the steps to solve it. They can use Pages or Notes to plan and write out their steps before entering the code.

Set up a help desk. Maintain a space where club experts can provide support to their peers.



Solve in multiple ways. Each puzzle has many solutions. If members finish early, encourage them to think of different ways to solve the puzzles. Thinking flexibly and comparing different solutions can help them improve their critical-thinking skills.

Pair programming. Have club members try working together on one iPad or Mac. They can brainstorm on how to solve the puzzles and take turns writing the code.

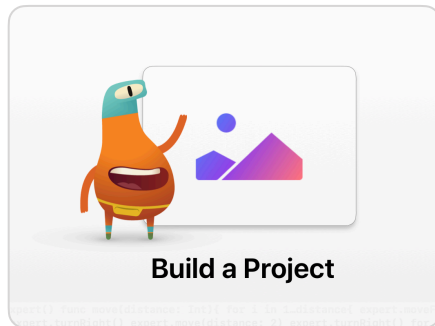
Use accessibility features. Swift Playgrounds works well with the built-in accessibility features in macOS or iPadOS so that everyone can learn to code. For example, coders can invert the colours, enable grayscale, and zoom to adjust visibility.



2. Choose your modules

Club materials are organized in modules that interweave coding and creative design activities. Each module consists of 12 one-hour sessions and addresses a particular theme and level of coding expertise. In Learn and Try sessions, club members explore key concepts and apply them to coding puzzles and challenges within Swift Playgrounds. And in Apply and Connect sessions, they consider how we use code to explore ideas and create new products. They apply their coding and design skills to build or design a project in Swift Playgrounds for a particular audience.

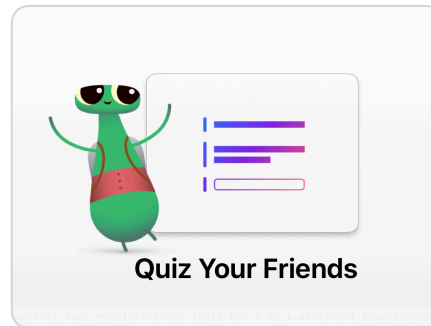
You'll find facilitator guides for each module in the second part of this document, or you can use the links below to explore them now.



Build a Project

Build a Project

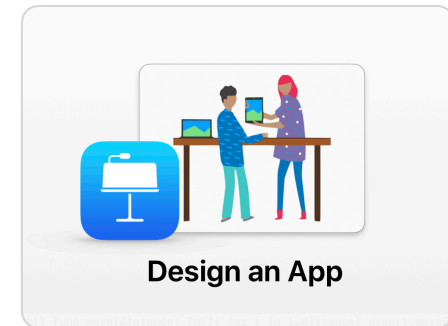
Club members master coding basics in Learn to Code 1 and Learn to Code 2 in Swift Playgrounds. They apply their new skills to design and build a playground that responds to touch events. [View module >](#)



Quiz Your Friends

Quiz Your Friends

Club members build on the skills they developed in Build a Project by completing more challenging puzzles in Learn to Code 1 and Learn to Code 2 in Swift Playgrounds. They create a playground that requests and responds to user information. [View module >](#)



Design an App

Design an App

Club members work together to design an app to help solve a problem in their community. They engage in a design process that shows them how to brainstorm, plan, prototype, and evaluate an app of their own. [View module >](#)



Celebrate

Community event or app showcase

Involve the broader community and explore the potential of code for solving contemporary problems by hosting a community event or app showcase. These events are also the perfect way to show off your club members' talents!

1. Plan the big event. Set a date and invite students, teachers, parents, and community members to attend.

Allow time for each team to present their project and hold a short Q&A session. If you have a large group, you can split the club into two rounds where members can watch each other's presentations.

Consider finishing the event with a fun slideshow of photos taken throughout club sessions.



2. Design awards. Friendly competition can be a great motivator. Inspire club members by offering awards that recognize specific strengths in coding and design — for example:

- Best Engineering
- Best Innovation
- Best Design
- Best Presentation

You could also encourage audience participation with a People's Choice award.



You can download and modify this [certificate](#) for different awards.



3. Recruit judges and mentors. Judges and mentors can be teachers or staff, students with expertise in coding, experts from the developer or design industry, members of the school board, local community leaders, or individuals who would benefit from the project idea.

Judges don't have to wait until the showcase to meet the club. Consider inviting them as guest speakers to share their expertise when learners are in the brainstorming or planning phase of their project.

4. Share and inspire. You may want to record the presentations. Share them with the broader community and create a highlight reel to inspire future club members.

Check out the App Showcase Guide for more inspiration and tips for App Showcase events.

[Download the App Showcase Guide >](#)





Everyone Can Code

Swift Coding Club

Certificate of Achievement

Awarded to

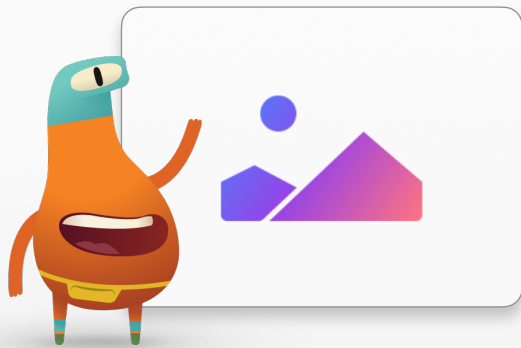
For



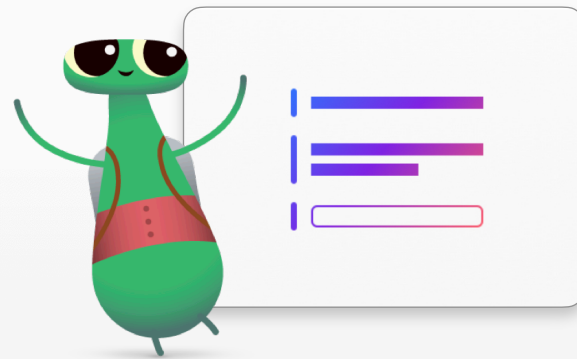
Signature

Date

Swift Coding Club Modules



Build a Project

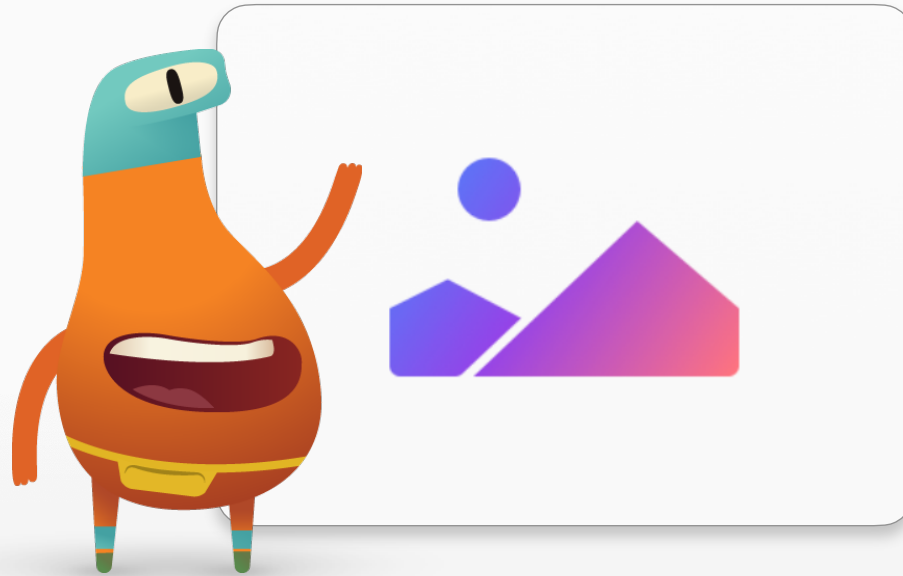


Quiz Your Friends



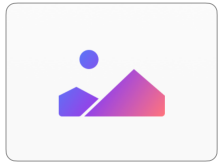
Design an App

```
func show(from actor: Actor) {guard state == .inactive else { return }state = .animatingToPicker // Use the role for the c
commandSpeed = WorldConfiguration.Actor.idleSpeedoriginalWorldActor = actororiginalActorTransform =
sceneNode.transformactor.reset()actor.sceneNode.runAction(.playSoundEffect(tapSource)) // Reset the existing 'pickerActors'.
d).for pickerActor in pickerActors {pickerActor.reset()pickerActor.isInCharacterPicker = true} let result = actor.perform
result.completionHandler = { _ inguard self.state == .animatingToPicker else TriggerDepthOfField(intro: true) { [unowned
oadAndDisplayCharacters()]} }func triggerDepthOfField(intro: Bool, completion: CompletionBlock? = nil) {let root = scene?.r
ode = root?.childNode(withName: "camera", recursively: true),let camera = cameraNode.camera else { return } SCNTransaction
saction.animationDuration = CharacterPickerController.fadeDurationcamera.focusDistance = intro ? focusDistanceMax : 0came
de : fStopDefault SCNTransaction.completionBlock = completionSCNTransaction.commit()}private func oadAndDisplayCharacters
return } / Setup overlay view & constraooverlayView.alpha = 0overlayView.autoresizingMask = [.flexibleWidth, .flexibleHeig
ldSubview(overlayView, view.bounds // Add a gesture recognizer to detect when character selection happens.let tapGesi
action: #selector(selectCharacter(:)) overlayView.gestureRecognizers = [tapGesture // Setup overlay sce
```



Build a Project

```
func show(from actor: Actor) {guard state == .inactive else { return }state = .animatingToPicker// Use the role for the c
commandSpeed = WorldConfiguration.Actor.idleSpeedoriginalWorldActor = actororiginalActorTransform =
sceneNode.transformactor.reset()actor.sceneNode.runAction(.playSoundEffect(tapSource)) // Reset the existing `pickerActors`.
d).for pickerActor in pickerActors {pickerActor.reset()pickerActor.isInCharacterPicker = true} let result = actor.perform
result.completionHandler = { _ inguard self.state == .animatingToPicker else TriggerDepthOfField(intro: true) { [unowned
oadAndDisplayCharacters()]}func triggerDepthOfField(intro: Bool, completion: CompletionBlock? = nil) {let root = scene?.r
ode = root?.childNodes(withName: "camera", recursively: true),let camera = cameraNode.camera else { return } SCNTransaction
saction.animationDuration = CharacterPickerController.fadeDurationcamera.focusDistance = intro ? focusDistanceMax : 0came
de : fStopDefault SCNTransaction.completionBlock = completionSCNTransaction.commit()}private func oadAndDisplayCharacters
return } / Setup overlay view & constraooverlayView.alpha = 0overlayView.autoresizingMask = [.flexibleWidth, .flexibleHeig
ldSubview(overlayView`view.bounds // Add a gesture recognizer to detect when character selection happens.let tapGesi
ction: #selector(selectCharacter( :)) overlayView.gestureRecognizers = [tapGesture // Setup overlay sce
```



Build a Project




Module Overview

In these sessions, club members master coding basics by completing fun activities from the Everyone Can Code Puzzles guide. They practice code by solving puzzles in Learn to Code 1 and Learn to Code 2 in Swift Playgrounds and apply their new skills to design and build a playground that responds to touch events.

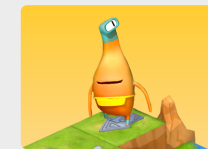
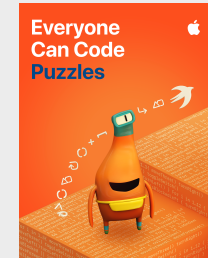
In Learn and Try sessions, club members explore key concepts and apply them to coding puzzles and challenges within Swift Playgrounds. In the Apply and Connect sessions, they'll learn about using code to explore ideas and create new products. At the end of the sessions, consider hosting a community event for club members to demonstrate their projects.

Page numbers from the Everyone Can Code Puzzles guide are included with the activities. To learn more about each activity, access additional resources, and find out how to support or challenge club members, explore the [Everyone Can Code Puzzles Teacher Guide](#).

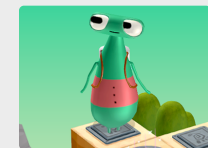
Session Overview

-  Learn and Try: 6 sessions
-  Apply and Connect: 6 sessions
-  Community Event

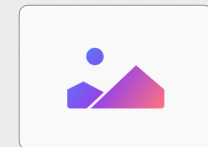
Resources



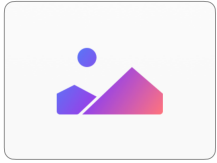
Learn to Code 1



Learn to Code 2



Shapes



Build a Project

1 Commands

Explore the foundational concept of a command — a specific instruction given to a computer. Learn to code using commands in a sequence.

Learn: Watch the introduction to Commands in Learn to Code 1

Hide and Seek (page 3)

Try: Complete puzzles in the Commands chapter in Learn to Code 1 (pages 4–10)

Learn to Code 1 Commands



- Introduction
- Issuing Commands
- Adding a New Command
- Toggling a Switch

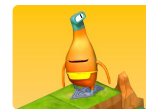
2 Functions

Learn how to create your own commands by authoring functions, and call functions you've written.

Learn: Watch the introduction to Functions in Learn to Code 1 Origami (page 15)

Try: Complete puzzles in the Functions chapter in Learn to Code 1 (pages 16–21)

Learn to Code 1 Functions



- Introduction
- Composing a New Behavior
- Creating a New Function
- Nesting Patterns

3 For Loops

Explore for loops and how you can make your code more efficient using functions and loops.

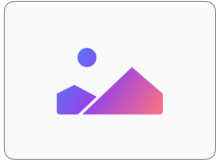
Learn: Watch the introduction to For Loops in Learn to Code 1 Pattern Maker (page 26)

Try: Complete puzzles in the For Loops chapter in Learn to Code 1 (pages 27–31)

Learn to Code 1 For Loops



- Introduction
- Using Loops
- Looping All the Sides



Build a Project

4 Variables

Learn about how computers store information using variables and how to code using variables.

Learn: Watch the introduction to Variables in Learn to Code 2

NewsBot (page 36)

Try: Complete puzzles in the Variables chapter in Learn to Code 1 (pages 37–40)

Learn to Code 2 Variables

- Introduction
- Keeping Track



5 Conditional Code

Explore Boolean logic and how to write conditional code.

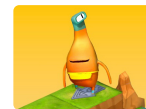
Learn: Watch the introduction to Conditional Code in Learn to Code 1

Someone Says (page 49)

Try: Complete puzzles in the Conditional Code chapter in Learn to Code 1 (pages 50–56)

Learn to Code 1 Conditional Code

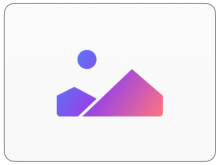
- Introduction
- Checking for Switches
- Using else if
- Looping Conditional Code
- Defining Smarter Functions



6 Design for an Audience

Consider different user perspectives and how to design products for a specific audience.

Connect: See it from someone else's point of view (page 58)



Build a Project

7 Types and Initialization

Learn how to describe types and how to initialize types in your code.

Learn: Watch the introductions to the Types and Initialization chapters in Learn to Code 2

Qualities of a Good Design (page 62)

Try: Complete puzzles in the Types and Initialization chapters in Learn to Code 2 (pages 63–66)

Learn to Code 2 Types

- Introduction
- Deactivating a Portal



Initialization

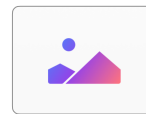
- Introduction
- Initializing Your Expert
- Using Instances of Different Types

8 Interactive Shapes

Explore the Shapes playground in the Books section of Swift Playgrounds, which is where you'll start your project development in the sessions that follow. Experiment with the Shape Graphics and Touches and Animations pages, and work out what each section of code achieves and how. As a group, list the graphic elements and functions available to you in the Shapes playground book.

Shapes

- Shape Graphics
- Touches and Animations



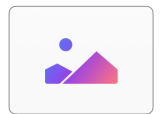
9 Build a Shapes Project

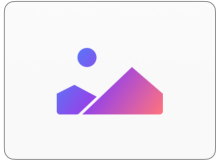
Explore how you can create a hand-eye coordination project in the Shapes playground book. Revisit and add to your list of graphic elements and functions.

Apply: Build a hand-eye coordination project (page 67)

Shapes

- Touches and Animations





Build a Project

10 Design a Project

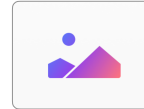
Brainstorm other projects you could create using the Shapes playground book. Consider the graphic elements and functions available and how they could meet the needs of a specific audience. Explore ideas collectively, then work in pairs to sketch an original idea that shows how the project achieves your purpose and is designed for a specific audience.

11 Build the Project

Work in pairs to code your project idea in the Touches and Animations page of the Shapes playground book. Refer to your sketch from the last session.

Shapes

- Touches and Animations

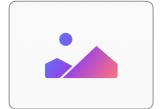


12 Evaluate the Project

Test your playground with your peers. Practice explaining how your project works — along with your design decisions — to prepare for the community event, where you'll share your creations.

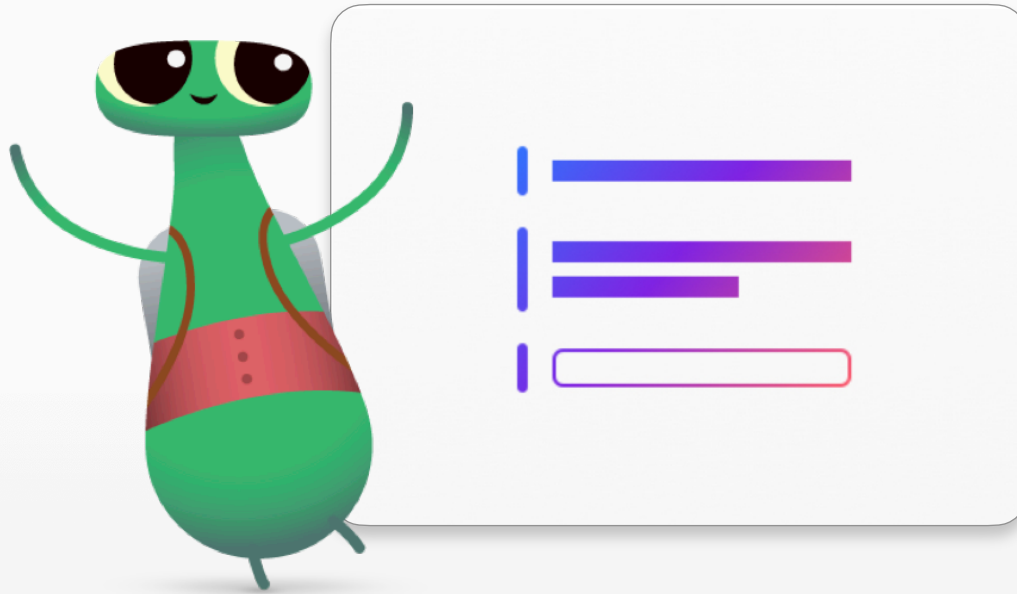
Shapes

- Touches and Animations



Community Event

Celebrate the club's achievements in a community event. You can demonstrate your project, explain your design process, and receive feedback from your community.



Quiz Your Friends

```
func show(from actor: Actor) {guard state == .inactive else { return }state = .animatingToPicker// Use the role for the c
commandSpeed = WorldConfiguration.Actor.idleSpeedoriginalWorldActor = actororiginalActorTransform =
sceneNode.transformactor.reset()actor.sceneNode.runAction(.playSoundEffect(tapSource)) // Reset the existing `pickerActors`.
d).for pickerActor in pickerActors {pickerActor.reset()pickerActor.isInCharacterPicker = true} let result = actor.perform
result.completionHandler = { _ inguard self.state == .animatingToPicker else TriggerDepthOfField(intro: true) { [unowned
oadAndDisplayCharacters()]}func triggerDepthOfField(intro: Bool, completion: CompletionBlock? = nil) {let root = scene?.r
ode = root?.childNodes(withName: "camera", recursively: true),let camera = cameraNode.camera else { return } SCNTransaction
saction.animationDuration = CharacterPickerController.fadeDurationcamera.focusDistance = intro ? focusDistanceMax : 0came
de : fStopDefault SCNTransaction.completionBlock = completionSCNTransaction.commit()}private func oadAndDisplayCharacters
return } / Setup overlay view & constraooverlayView.alpha = 0overlayView.autoresizingMask = [.flexibleWidth, .flexibleHeig
ldSubview(overlayView)`view.bounds // Add a gesture recognizer to detect when character selection happens.let tapGesi
ction: #selector(selectCharacter( :)) overlayView.gestureRecognizers = [tapGesture // Setup overlay sc
```



Quiz Your Friends




Module Overview

In this module, club members build on their skills by completing more challenging activities in the Everyone Can Code Puzzles guide. They practice their code by solving puzzles in Learn to Code 1 and Learn to Code 2 in Swift Playgrounds and use their advanced skills to develop a playground that requests and responds to user information. This module requires an understanding of materials in Puzzles chapters 1–6, the completion of the Build a Project clubs module, or equivalent background knowledge.

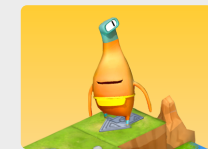
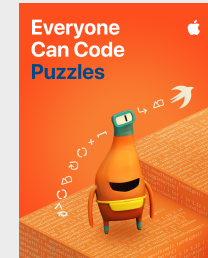
In Learn and Try sessions, club members explore key concepts and apply them to coding puzzles and challenges within Swift Playgrounds. In the Apply and Connect sessions, they'll learn about using code to explore ideas and create new products. At the end of the sessions, consider hosting a community event for club members to demonstrate their projects.

Page numbers from the Everyone Can Code Puzzles guide are included with the activities. To learn more about each activity, access additional resources, and find out how to support or challenge club members, explore the [Everyone Can Code Puzzles Teacher Guide](#).

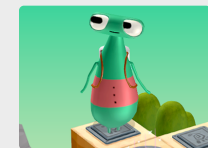
Session Overview

-  Learn and Try: 4 sessions
-  Apply and Connect: 8 sessions
-  Community Event

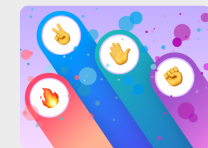
Resources



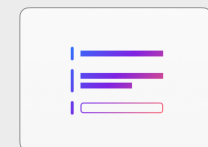
Learn to Code 1



Learn to Code 2



Rock Paper Scissors



Answers



Quiz Your Friends

1 Functions with Parameters

Learn how to give computers more information by making functions more specific with parameters.

Learn: Watch the introduction to Parameters in Learn to Code 2

Recipe for Success (page 71)

Try: Complete puzzles in the Parameters chapter in Learn to Code 2 (pages 72–75)

Learn to Code 2 Parameters

- Introduction
- Moving Further Forward



2 Design a Game

Use the Rock, Paper, Scissors playground in the Books section of Swift Playgrounds to design a new and improved edition of the game.

Apply: Build a Rock, Paper, Scissors game (page 76)

Rock, Paper, Scissors

- Creating a Game
- Sharing Code
- Adding Actions
- Modifying Properties



3 Logical Operators

Learn how to code specific behaviour in response to certain conditions using logical operators.

Learn: Watch the introduction to Logical Operators in Learn to Code 1

Someone Says, Round 2 (page 81)

Try: Complete puzzles in the Logical Operators chapter in Learn to Code 1 (pages 82–85)

Learn to Code 1 Logical Operators

- Introduction
- Using the NOT Operator
- Checking This AND That
- Checking This OR That





Quiz Your Friends

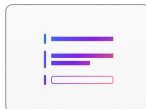
4 Create a Quiz

Combine knowledge of operators with conditions, variables, functions, and functions with parameters to create a quiz in the Answers playground in the Books section of Swift Playgrounds.

Apply: Build a quiz (page 86)

Answers

- Text
- API Overview
- Which Type Are You Quiz



5 Design a Quiz Project

Come up with an idea for your own playground quiz, built on the Answers playground book. Determine the purpose of your quiz, explore designs for quiz apps, consider the target audience, and sketch your own idea.

6 While Loops

Learn about while loops and how to use them to loop a block of code until a condition is true.

Learn: Watch the introduction to While Loops in Learn to Code 1

Playground Games (page 90)

Try: Complete the puzzles in the While Loops chapter of Learn to Code 1 (pages 91–94)

Learn to Code 1 While Loops

- Introduction
- Running Code While...
- Create Smarter While Loops
- Nesting Loops





Quiz Your Friends

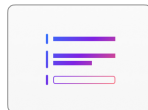
7 Refine the Quiz

Update your original quiz to bring different modes into your while loops. You'll use these skills in later sessions when you code your own project idea.

Apply: Refine your quiz (page 95)

Answers

- Text
- API Overview
- Which Type Are You Quiz



8 Arrays and Refactoring

In this session, club members learn new technical skills using arrays, then use those skills to refactor their code.

Learn: Watch the introduction to Arrays in Learn to Code 2

Evaluate (page 99)

Try: Complete the puzzles in the Arrays chapter in Learn to Code 2 (pages 100–105)

Learn to Code 2 Arrays

- Introduction
- Storing Information
- Iteration Exploration
- Stacking Blocks
- Getting in Order
- Fixing Index Out of Range Errors



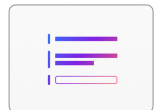
9 Add Choices to the Quiz

Update your quiz playground to include lists of choices, and begin to imagine the projects you might build with lists of choices.

Apply: Add choices to your quiz (page 106)

Answers

- Text
- API Overview
- Which Type Are You Quiz





Quiz Your Friends

10 Design a new Project

Brainstorm other projects you could create using the Answers playground book. Explore ideas collectively, then work independently to come up with an idea, identify the purpose and audience, and sketch a wireframe.

11 Build the Project

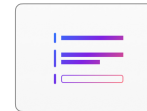
Build your own project in the Answers playground book. Use your wireframe from the previous session as a guide.

12 Evaluate the Project

Test your playground with your peers. Practice explaining how your project works — along with your design decisions — to prepare for the community event, where you'll share your creations.

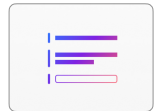
Answers

- Text
- API Overview
- Which Type Are You Quiz



Answers

- Text
- API Overview
- Which Type Are You Quiz



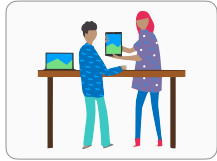
Community Event

Celebrate the club's achievements in a community event. You can demonstrate your project, explain your design process, and receive feedback from your community.



Design an App

```
func show(from actor: Actor) {guard state == .inactive else { return }state = .animatingToPicker// Use the role for the c
commandSpeed = WorldConfiguration.Actor.idleSpeedoriginalWorldActor = actororiginalActorTransform =
sceneNode.transformactor.reset()actor.sceneNode.runAction(.playSoundEffect(tapSource)) // Reset the existing `pickerActors`.
d).for pickerActor in pickerActors {pickerActor.reset()pickerActor.isInCharacterPicker = true} let result = actor.perform
result.completionHandler = { _ inguard self.state == .animatingToPicker else TriggerDepthOfField(intro: true) { [unowned
oadAndDisplayCharacters()}}}func triggerDepthOfField(intro: Bool, completion: CompletionBlock? = nil) {let root = scene?.r
ode = root?.childNode(withName: "camera", recursively: true),let camera = cameraNode.camera else { return } SCNTransaction
saction.animationDuration = CharacterPickerController.fadeDurationcamera.focusDistance = intro ? focusDistanceMax : 0came
de : fStopDefault SCNTransaction.completionBlock = completionSCNTransaction.commit()}private func oadAndDisplayCharacters
return } / Setup overlay view & constraooverlayView.alpha = 0overlayView.autoresizingMask = [.flexibleWidth, .flexibleHeig
ldSubview(overlayView)`view.bounds // Add a gesture recognizer to detect when character selection happens.let tapGesi
ction: #selector(selectCharacter(-:)) overlayView.gestureRecognizers = [tapGesture // Setup overlay sce
```



Design an App

Module Overview

In this module, club members work in small teams to design an app to help solve a problem in their community. They're guided through a design process in which they brainstorm ideas, plan their app, build a working prototype in Keynote, and evaluate the app.

The design process content is presented in an [App Design Journal](#) that helps club members record and track their ideas as they go through the design cycle. The idea is to document their process to help reiterate and improve their app project. It's also very useful as a reference and starting point for future projects.

At the end of this module, host an App Showcase to celebrate your club members' ingenuity. Download the [App Showcase Guide](#) for tips and resources to plan your event.

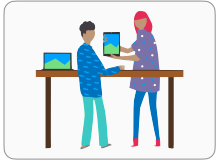
Session Overview

- **Brainstorm: 2 sessions**
- **Plan: 2 sessions**
- **Prototype: 5 sessions**
- **Evaluate: 3 sessions**
- **Showcase**

Resources



App Design Journal



Design an App

1–2 Brainstorm

Explore app ideas to address something you care about, and focus on the purpose and audience of your app.

Brainstorm

- Purpose
- Ideas
- Focus



3–4 Plan

Consider what your app will actually do to achieve the purpose while you explore iOS features and inclusive practices.

Plan

- User Actions
- Input and App State
- Choose Features
- Inclusion



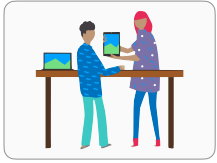
5–9 Prototype

Design your app's user interface, storyboard your screens, and build a working prototype of your app in Keynote.

Prototype

- Sketch Screens
- Storyboard
- Refine App Behaviour
- Design Style
- Build
- App Icon and Name





Design an App

10–12 Evaluate

Pitch your app idea, and have users test your prototype. Then iterate on your design in response to feedback.

Evaluate

- App Pitch
- Prepare
- Observation



App Showcase

Share the club's app prototypes and pitches with the broader community through an App Showcase. Find inspiration for planning and conducting your event in the [App Showcase Guide](#).



© 2024 Apple Inc. All rights reserved. Apple, the Apple logo, iPad, iPadOS, Keynote, Mac, macOS, Pages, Swift, Swift Logo, Swift Playgrounds, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries. Everyone Can Code is a service mark of Apple Inc, registered in the U.S. and other countries. IOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license. Other product and company names mentioned herein may be trademarks of their respective companies. January 2022