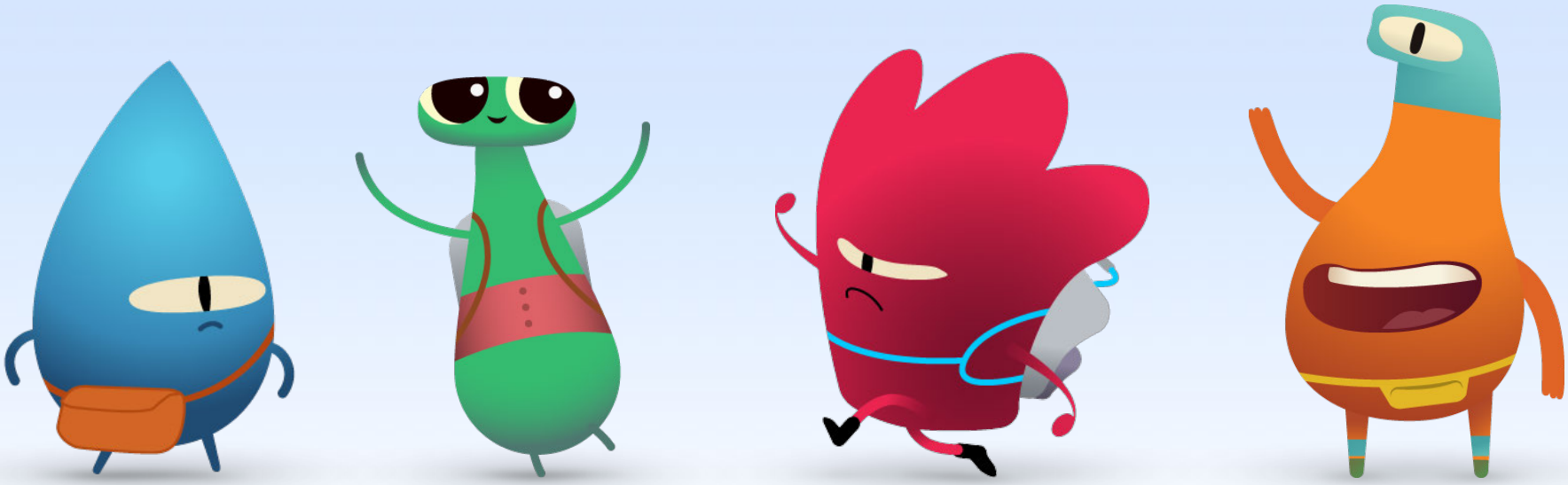
Swift Coding Club



Swift Playgrounds Kit

Welcome to the Swift Coding Club!

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.

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 kit is arranged in three sections:

Get Started

Everything you need to launch a Swift Coding Club.

Learn & Design

Tips and activities for designing club sessions.

Celebrate

Helpful resources to plan and host an app showcase in your community.

Swift Coding Clubs

Block-Based Coding | Ages 8–11

Learn coding basics using visual apps on iPad.

Swift Playgrounds | Ages 11+

Use Swift code to learn coding fundamentals with Swift Playgrounds on iPad.

Xcode | Ages 14+

Learn to develop apps in Xcode on Mac.

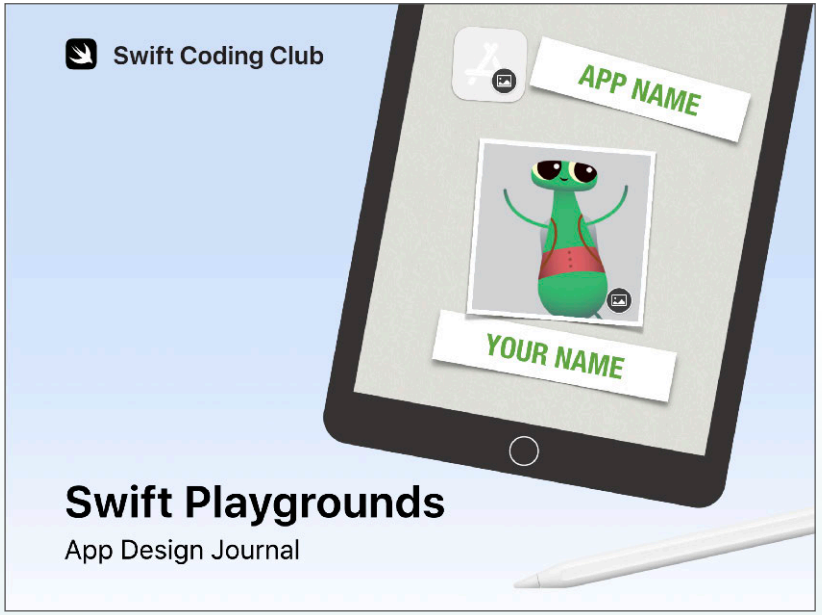
Get Started

1. Download club materials.

Use AirDrop to share these two guides with club members in your first club meeting. They're also included as part of this document.

Coding Activities
Learn coding concepts with these fun, collaborative activities and solve puzzles with the Swift Playgrounds app on iPad.

Download Swift Playgrounds Coding Activities >



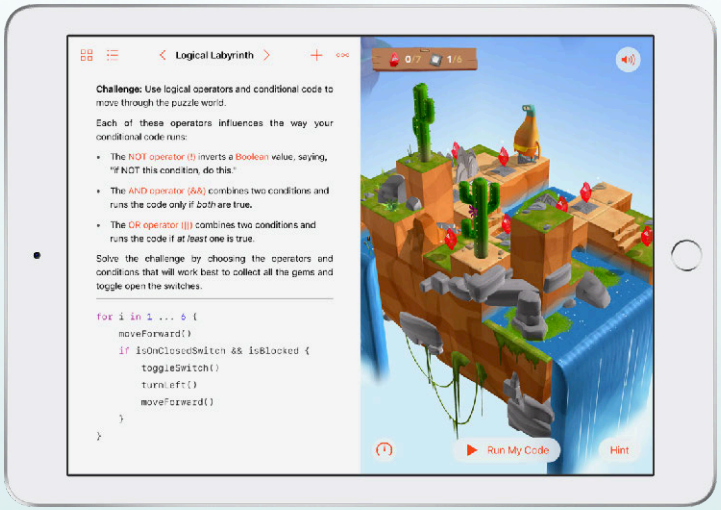
App Design Journal
Explore the app design process with this Keynote journal. Brainstorm, plan, prototype, and evaluate your club's app ideas.

Download Swift Playgrounds App Design Journal >

2. Check your tech.

Before your first meeting, be sure you have the following:

- **iPad.** iPad mini 2 or later, iPad Air or later, or iPad Pro running iOS 11 or later. It's best if each person has their own device, but they can also share and code together.
- **Swift Playgrounds app.** Download Swift Playgrounds >
- **Learn to Code 1 and 2 playgrounds**. Download these playgrounds from within the Swift Playgrounds app.
- **Keynote.** You'll use the Keynote app on iPad for your app prototypes.
- **Swift Coding Club materials.**



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?

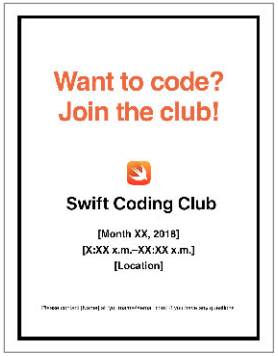
4. Spread the word.

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 what types of apps they'd want to create. Talk about ideas for holding an app design showcase and how 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

Learn & Design

The club materials are designed for you to interweave coding and app design activities. You can also add sessions that support your members' interests. Below is a sample schedule for 30 one-hour club sessions.

Sessions 1–5

- Setup
- Coding Activities 1 & 2
- App Design Journal: Brainstorm

Sessions 6–10

- Coding Activities 3 & 4
- App Design Journal: Plan

Sessions 11–15

- Coding Activity 5
- App Design Journal: Prototype

Sessions 16–20

- Coding Activity 6
- App Design Journal: Prototype

Sessions 21–25

- Coding Activity 7
- App Design Journal: Evaluate and Reiterate

Sessions 26–30

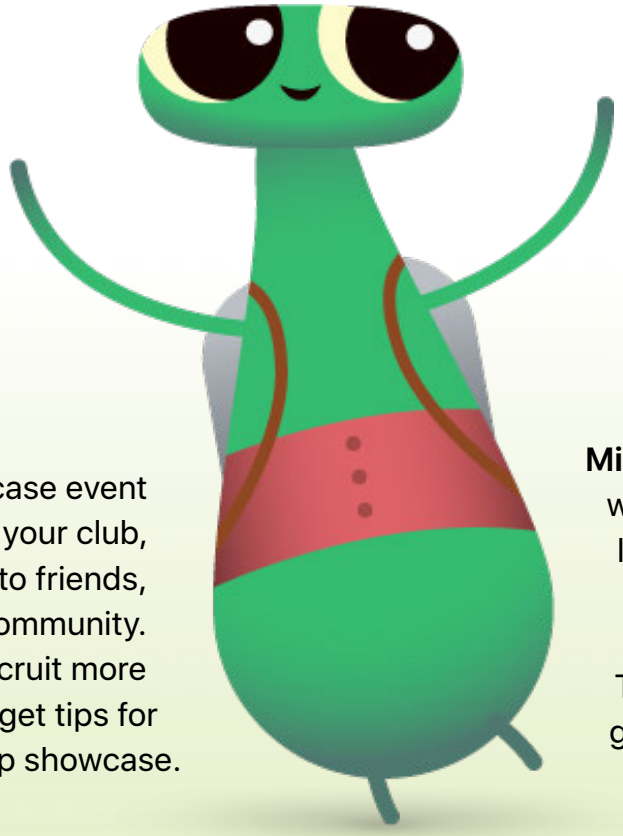
- App Design Journal: App Pitch
- App Design Showcase

Consider adding sessions to expand on app design and coding activities, like building a drone obstacle course or creating a robot rescue mission challenge. To prompt app design brainstorming, you might even want to add guest speakers or field trips.

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.

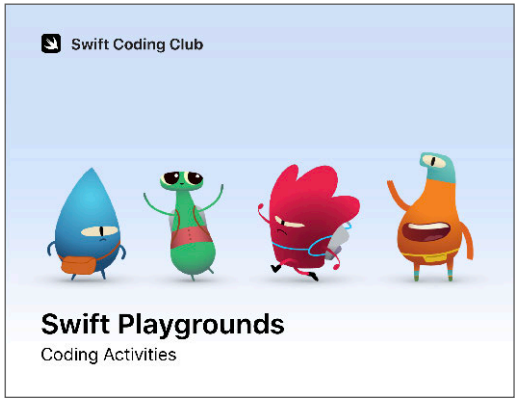


Share ideas. Some members will be interested in making games. Others might want to create apps to help people, learn Swift, or control robots. Think about ways for members to work together on projects they care about.

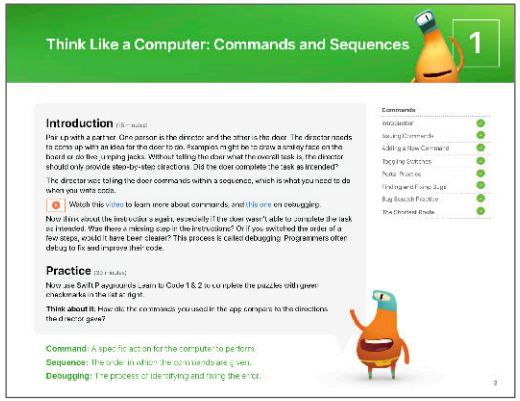
Show off. An app showcase event is a great way to promote your club, app ideas, and coding skills to friends, families, teachers, and the community. It might even help you recruit more members. See page 13 to get tips for holding your own app showcase.

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!

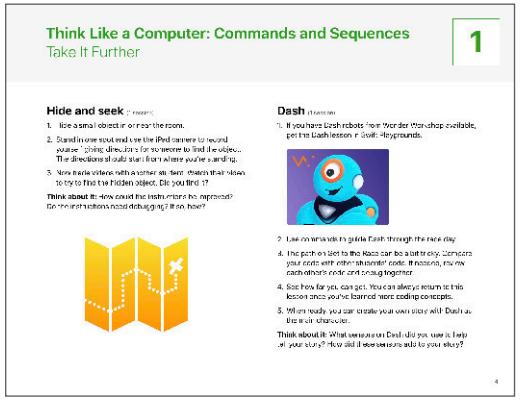
Swift Playgrounds Coding Activities



Coding activities: Built around Swift Playgrounds, these collaborative activities introduce fundamental coding concepts and skills.

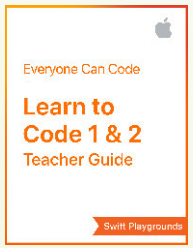


Coding concepts: In each activity, club members will learn about a fundamental coding concept and explore it in an everyday context. They'll then apply the coding concept to solve puzzles in Swift Playgrounds.



Take It Further: Each coding concept has two Take It Further activities. The first activity deepens understanding of the coding concept and fosters communication and teamwork. Members use iPad to apply their understanding in a creative project.

The second optional activity challenges members to apply the concept in a playground from the Challenges, Starting Points, and Subscriptions sections of Swift Playgrounds. Some activities require specific connected devices.



Need more information or want to go deeper?

Download Swift Playgrounds: Learn to Code 1 & 2 Teacher Guide >

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.

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.

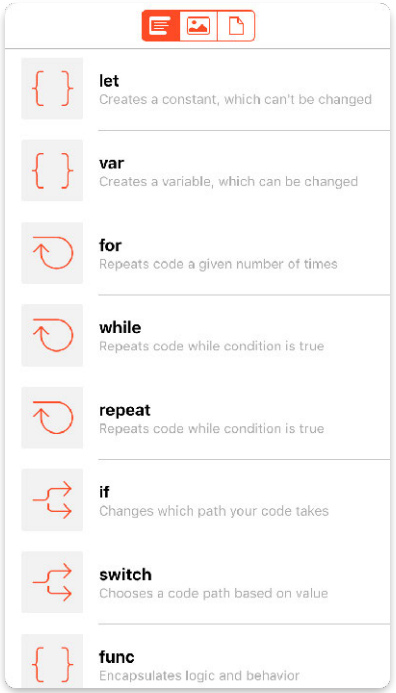
Break down the puzzles. The puzzles get tricky. Club members can divide a puzzle into parts to help them think through all the steps to solve it. They can use Pages or Notes to plan and write out their steps before entering the code.

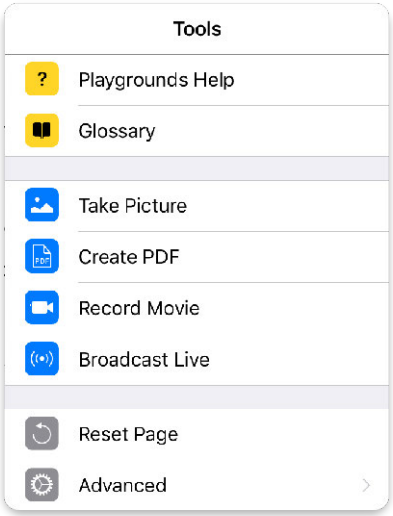
Pair coding. Have club members try working together on one iPad. They can brainstorm on how to solve the puzzles and take turns writing the code.

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

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

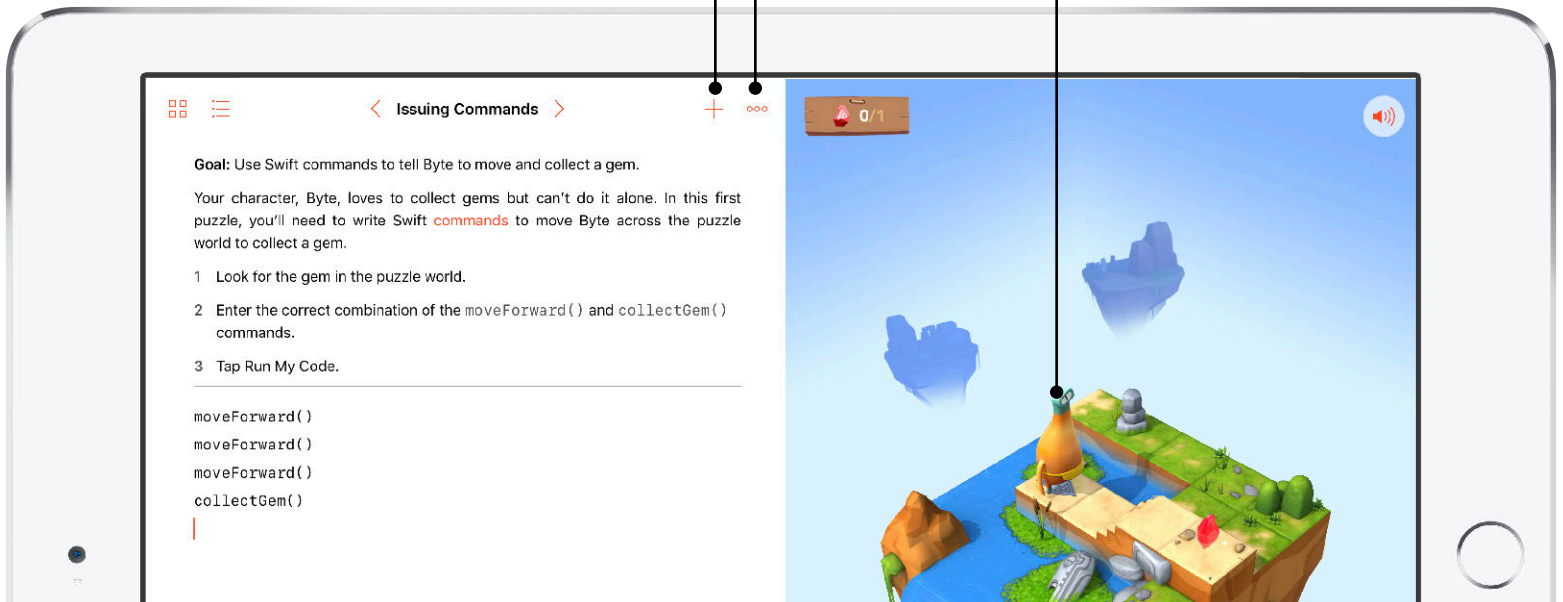
Explore Swift Playgrounds

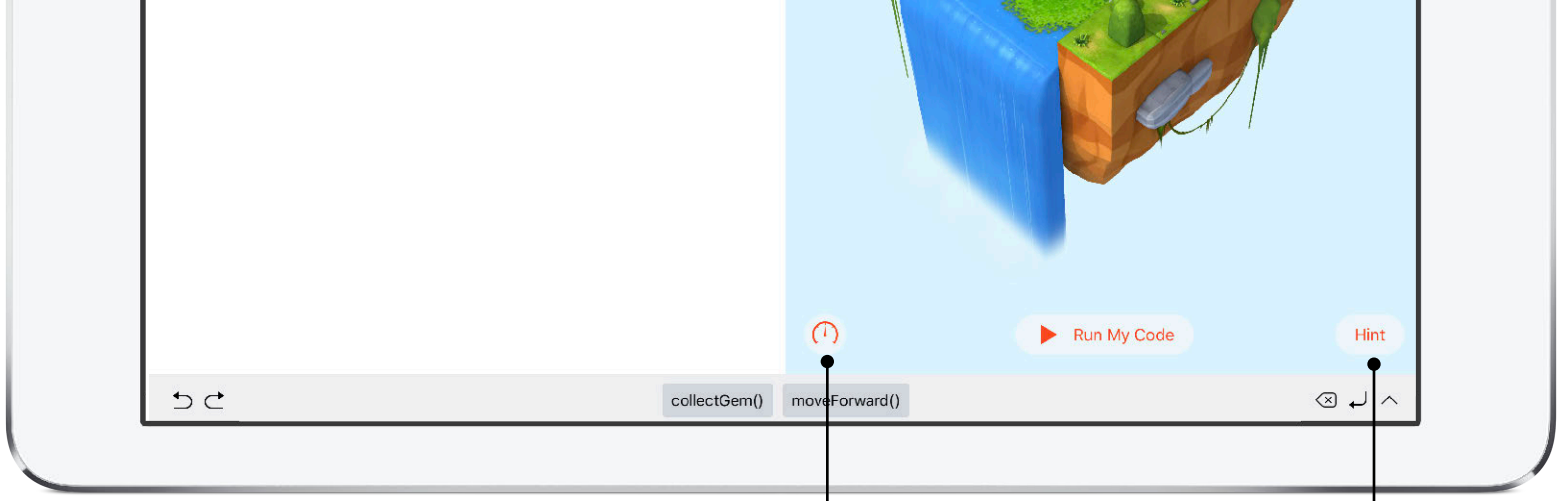


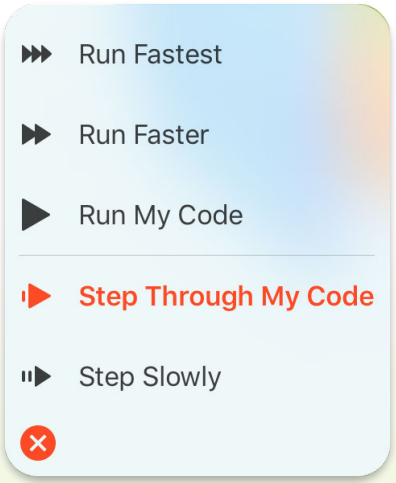


Snippets Library. To minimize typing, tap in the toolbar to access the Snippets Library and quickly drag commonly used pieces of code.

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





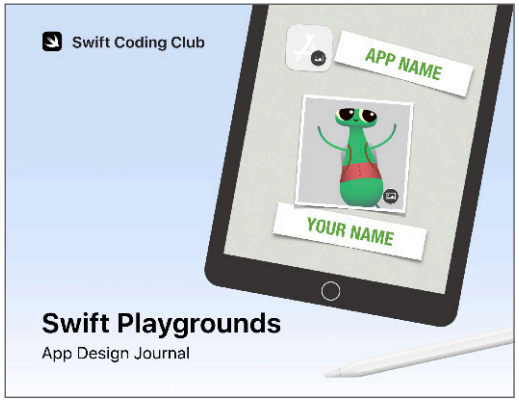


Control the speed. Speed up or slow down the code.

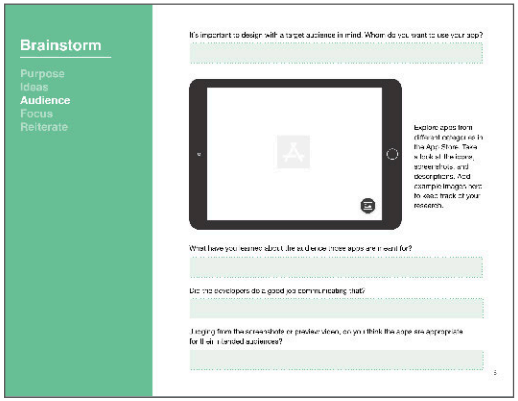
Highlight code as it runs. Use Step Through My Code to highlight each line of code as it runs to better understand what the code is doing.

Hint. This feature provides suggestions to help learners. It also reveals a puzzle's solution eventually, though coders can't simply cut and paste the solution. To move on, they still have to complete the steps and write the code themselves.

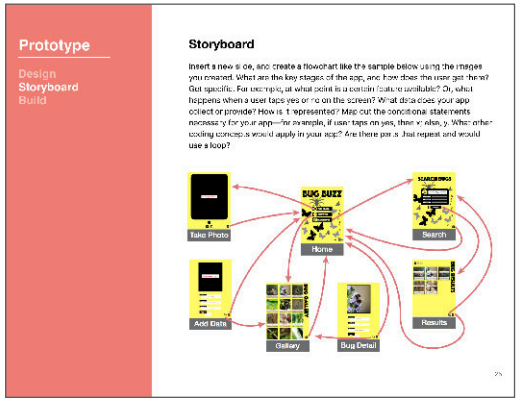
App Design Journal



Coders use this Keynote journal to learn about app features and design an app to solve a community problem.



Club members work in small teams to brainstorm and plan the app solution, then create a working prototype of the app in Keynote.



The journal walks coders through the process of evaluating their designs and iterating on their prototypes—just like professional app designers.

Members create a three-minute app pitch presentation or video and celebrate their work in an app design showcase.

Need Keynote tips?

Download Keynote for iPad Starter Guide iOS 11 >

Celebrate

App design showcase

The app design process and the showcase are powerful opportunities to involve the broader community and explore the potential of apps for solving contemporary problems. The showcase is also the perfect way to show off the talents of your club members!

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

Allow time for each team to present their app pitch and to 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 pitches.

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 app design. Consider awards for:

- Best Engineering
- Best Innovation
- Best Design
- Best Pitch

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

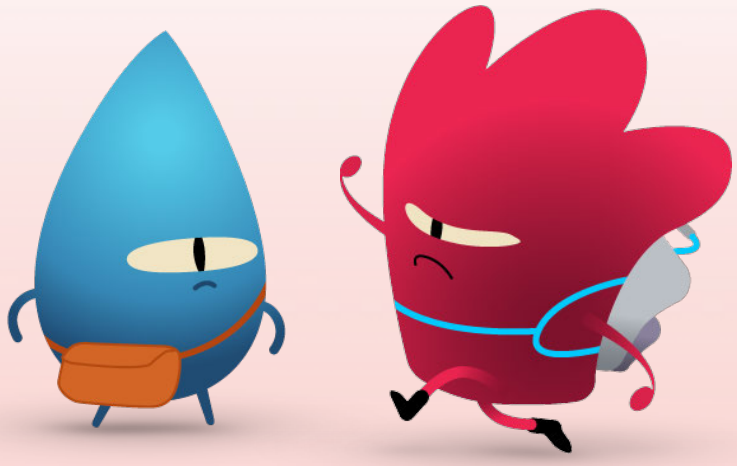
You can download and modify this certificate for different awards.

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 app 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 app design.

4. Pick a winner. Judges can use the rubric on the next page to help them evaluate the app pitches and provide feedback. You could also share the rubric with coders before the showcase as part of the evaluation phase of the app design process.

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



Evaluation Rubric

Download >

	Novice (1 point)	Intermediate (2 points)	Proficient (3 points)	Mastered (4 points)	Points
Pitch Content	The pitch shares key information about the app, such as its purpose and target audience.	The pitch clearly explains how the app was designed and how it improves on existing apps that have a similar purpose and audience.	The pitch explains how the app was designed to meet a market demand and how its solution is unique.	The pitch provides evidence of market demand and explains how the app design process ensures app success with key stakeholders.	
Pitch Delivery	The team explains key points during their pitch.	The team delivers their pitch with confidence and enthusiasm.	The team delivers an engaging pitch using a range of audience engagement techniques.	The pitch is well articulated, creative, memorable, and fluid across the team.	
User Interface	The UI design complements the purpose of the app and has thematically consistent screens.	The UI design applies familiar iOS interface elements, icons, and text styles to achieve clarity and function. The prototype has interactive elements that demonstrate the app's behavior.	The UI is elegant, concise, and pleasing, with attention given to color, layout, and readability. It gives feedback to the user about their progress through the app, or the options available to them when they perform an action.	The app design defers to its content as the most important element. The UI design empowers the user to directly interact with and manipulate its content. It uses animation to provide additional feedback for interactions.	
User Experience	The prototype expresses a clear intent for the app and how users can interact with it to accomplish a goal.	The prototype uses consistent and standard navigation techniques to provide the user with a clear and intuitive path through its content.	The prototype enables users to view and interact with its content differently depending on their needs. The prototype addresses accessibility and includes features to protect user privacy or online safety.	The prototype innovates on best practices of similar apps and caters to the needs of both new and experienced users. It exhibits a style and personality that set it apart from its peers.	
Coding Concepts	The team describes how their app design would relate to its code, such as the kind of data it stores or how it reacts to different user inputs.	The team explains how basic coding concepts like data types, conditional logic, and touch events relate to the design of their app.	The team describes specific coding tasks that would be necessary to implement their app, as well as how that code powers its screens and/or interactions.	The team explains the algorithms at the heart of their app, and describes them conversationally or in pseudocode. They describe the app's functional parts and its data, and how they're structured and interrelated.	
Comments:					**Total score**

Swift Coding Club

Swift Playgrounds

Certificate of Achievement

Awarded to

For

Signature

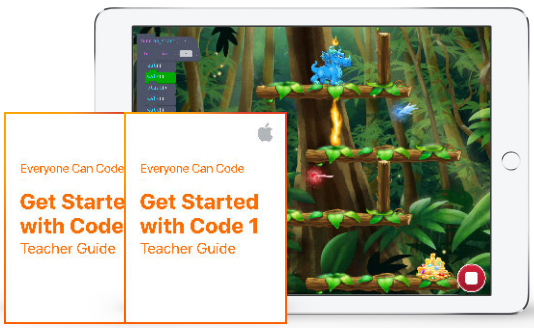
Date

Take It Further

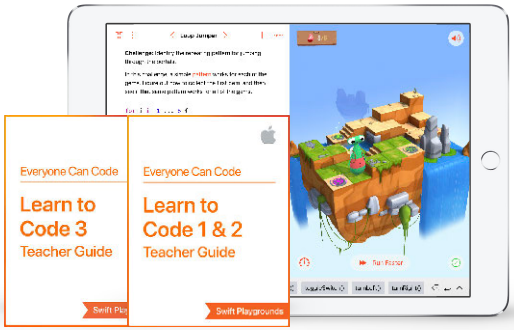
Swift Coding Club is just the beginning of your coding journey. The Everyone Can Code curriculum provides fun, supportive resources to take coders from learning the basics on iPad to building real apps on Mac.

And you don't have to stop at club activities. Comprehensive Teacher Guides also enable teachers to bring coding into the classroom, with step-by-step, curriculum-aligned lessons for students from kindergarten to college.

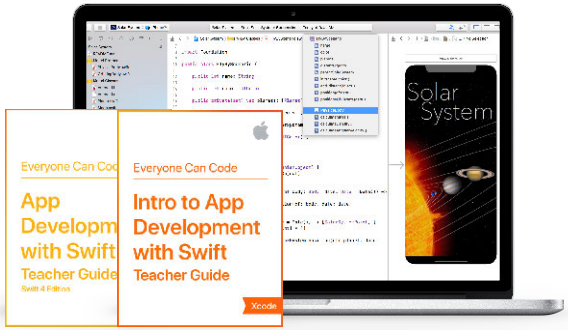
See all the Everyone Can Code resources >



Learn more about the Get Started with Code curriculum >

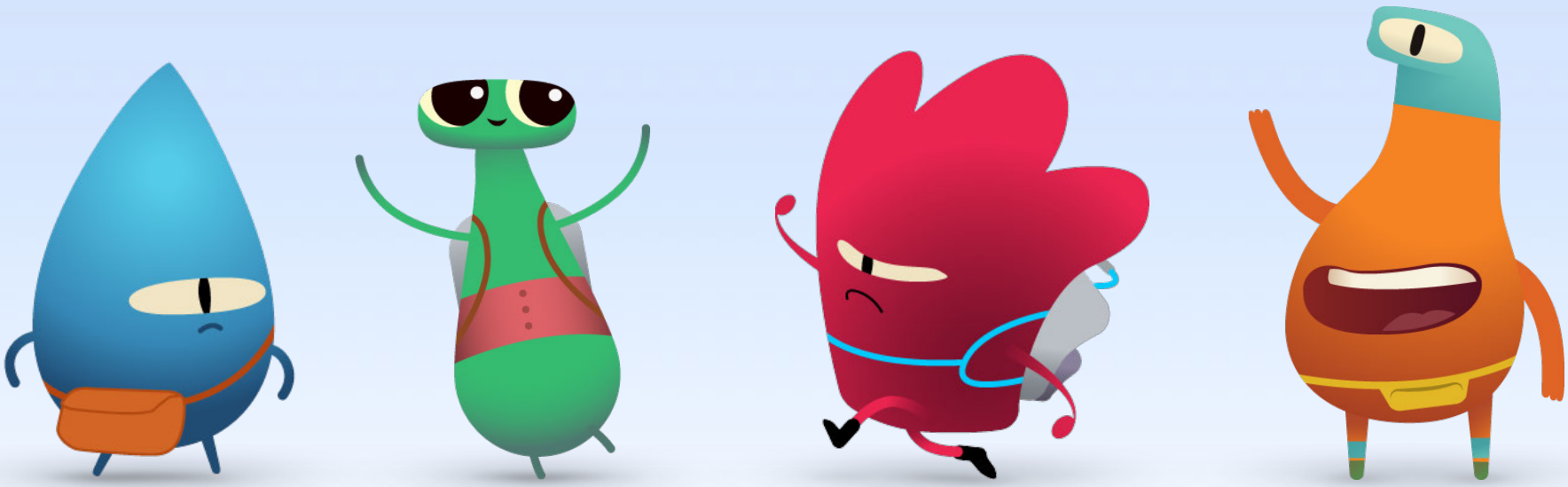


Learn more about the Swift Playgrounds curriculum >



Learn more about the App Development with Swift curriculum >

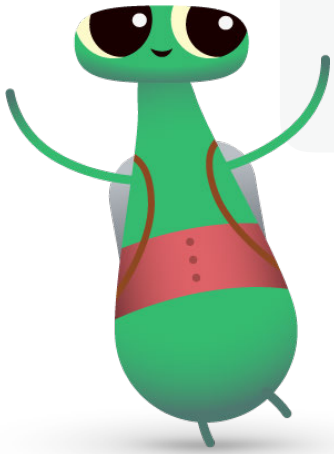
Swift Coding Club



Swift Playgrounds

Coding Activities

Topics



Welcome to the Swift Coding Club!

These coding activities cover fundamental coding concepts to go along with the app design activities. With these activities, you'll not only build up your coding skills, but you'll start to understand how apps work. This will help you design better apps.

For each topic, you'll learn about a specific coding concept with a brief introduction activity, then you'll apply that coding concept to solve puzzles in Swift Playgrounds.

There are also "Take It Further" activities for each topic to help you learn more about that concept. These additional activities are optional and you can choose to do none, one, or both activities. Note that some of these activities require another device like a robot or drone. If you have these devices, they're a great way to apply what you've learned.

Have fun!

Think Like a Computer: Commands and Sequences

1

Introduction (15 minutes)

Pair up with a partner. One person is the director and the other is the doer. The director needs to come up with an idea for the doer to do. Examples might be to draw a smiley face on the board or do five jumping jacks. Without telling the doer what the overall task is, the director should only provide step-by-step directions. Did the doer complete the task as intended?

The director was telling the doer commands within a sequence, which is what you need to do when you write code.

Watch this video to learn more about commands, and this one on debugging.

Now think about the instructions again, especially if the doer wasn't able to complete the task as intended. Was there a missing step in the instructions? Or if you switched the order of a few steps, would it have been clearer? This process is called debugging. Programmers often debug to fix and improve their code.

Practice (30 minutes)

Now use Swift Playgrounds Learn to Code 1 to complete the puzzles with green checkmarks in the list at right.

Think about it: How did the commands you used in the app compare to the directions the director gave?

Commands

- Introduction ✓
- Issuing Commands ✓
- Adding a New Command ✓
- Toggling Switches ✓
- Portal Practice ✓
- Finding and Fixing Bugs ✓
- Bug Squash Practice ✓
- The Shortest Route ✓

Command: A specific action for the computer to perform.

Sequence: The order in which the commands are given.

Debugging: The process of identifying and fixing the error.

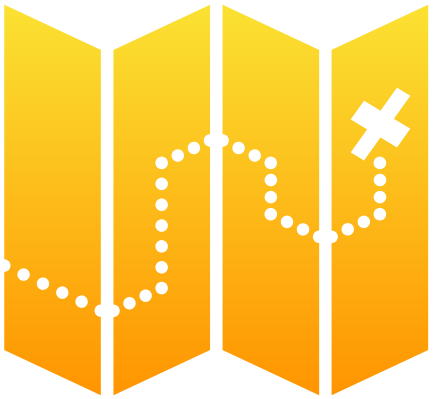
Think Like a Computer: Commands and Sequences
Take It Further

1

Hide and seek (1 session)

1. Hide a small object in or near the room.
2. Stand in one spot and use the iPad camera to record yourself giving directions for someone to find the object. The directions should start from where you're standing.
3. Now trade videos with another student. Watch their video to try to find the hidden object. Did you find it?

Think about it: How could the instructions be improved? Do the instructions need debugging? If so, how?



Dash (1 session)

1. If you have Dash robots from Wonder Workshop available, get the Dash lesson in Swift Playgrounds.

2. Use commands to guide Dash through the race day.
3. The path on Get to the Race can be a bit tricky. Compare your code with other students' code. If needed, review each other's code and debug together.
4. See how far you can get. You can always return to this lesson once you've learned more coding concepts.
5. When ready, you can create your own story with Dash as the main character.

Think about it: What sensors on Dash did you use to help tell your story? How did these sensors add to your story?

Think Efficiently: Functions and a Bit of Loops

2

Introduction (5 minutes)

Think of some dance moves. Describe the moves to each other, or better yet, get others to perform them. How easy was it to describe?

In programming, it's sometimes easier to combine existing commands to create a new behavior. This process is called composition. When you name the new behavior so you can use it again in the future, you've created a function. When you tell a program to run a function, you're "calling" it. So if somebody said, "Do the Macarena," they're calling the function "Macarena."

Take a look at this video on functions and loops.

Practice (40 minutes)

Now use Swift Playgrounds Learn to Code 1 to complete the puzzles with green checkmarks in the list at right.

Think about it: In a given puzzle, how many moves did the character make? And how many commands did you write? When and why should you create functions and loops?

Function: A collection of commands grouped together and given a name.

For loop: Runs a block of code over and over for a set number of times.

Functions

- Introduction ✓
- Composing a New Behavior ✓
- Creating a New Function ✓
- Collect, Toggle, Repeat
- Across the Board
- Nesting Patterns ✓
- Slotted Stairways ✓
- Treasure Hunt

For Loops

- Introduction ✓
- Using Loops ✓
- Looping All the Sides ✓
- To the Edge and Back
- Loop Jumper
- Branch Out
- Gem Farm
- Four Stash Sweep

Think Efficiently: Functions and a Bit of Loops

Take It Further

2

Pattern maker (1 session)

1. Create a pattern using a drawing app like Art Set or Pages, incorporating different shapes, objects, and colors. The pattern can be as long as you'd like.
2. In the app, write your pattern 20 times with words—for example, "red, yellow, blue; red, yellow, blue…" and so on.
3. Create a name for the part of the pattern that repeats (for example, red, yellow, blue = Primary Colors), then write the pattern again in words using only the new name.
4. How much less work or fewer steps did it take to write the pattern?
5. How many times does Primary Colors repeat? Now describe your pattern in one step. You've written a for loop!

Think about it: How does this activity relate to coding?

Robot interview (1 session)

1. Get the Answers Starting Point in Swift Playgrounds.

2. On the Text page, "show" and "ask" are functions. Tap Run My Code and fill in your name, then tap Submit to see what happens. Functions can have a result, which is what you see in the live view.
3. On the Types page, explore different "show" and "ask" functions.
4. Pair up with a club mate and write a series of different "show" and "ask" functions for each other to complete.
5. Use the results from your functions to write a fictional story, an interview article, or a short biography.

Think about it: What if you wrote your "show" and "ask" functions in a different sequence? How would it affect your story or interview?

Think Logically: Conditional Code

3

Introduction (10 minutes)

As a group, play a couple rounds of the I Spy game. In the game, a spy chooses an object, then describes only one part of it to the players. The players then have to look around their environment and guess what the spy saw. A student who guesses correctly gets to be the next spy.

What decisions did you have to make? What was your thought process when trying to figure out what the spy saw? To play I Spy, you were thinking in terms of conditions.

Watch this video to learn more about conditional code.

Practice (35 minutes)

Now use Swift Playgrounds Learn to Code 1 to complete the puzzles with green checkmarks in the list at right.

Think about it: What kinds of decisions did your code make using the if statement? How did you combine for loops and if statements? Why?

Condition: Something you test that results in true or false.

Conditional code: A block of code that will run only if something is true.

Boolean: A value that can only be either true or false.

Logical operator: A symbol or words like “and,” “or,” and “not.”

Conditional Code

- Introduction ✓
- Checking for Switches ✓
- Using else if ✓
- Looping Conditional Code ✓
- Conditional Climb
- Defining Smarter Functions ✓
- Boxed In
- Decision Tree

Logical Operators

- Introduction ✓
- Using the NOT Operator ✓
- Serial of NOT
- Checking This AND That ✓
- Checking This OR That ✓
- Logical Labyrinth

Think Logically: Conditional Code
Take It Further

3

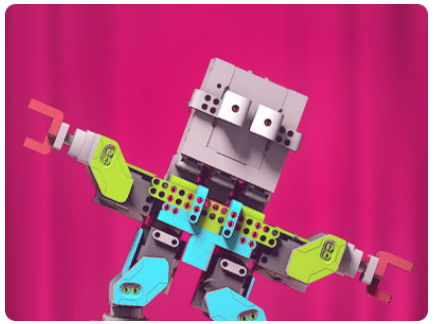
Scavenger hunt (1 session)

1. Each student should write two conditions on separate pieces of paper—for example, "Is a rectangle" or "Starts with the letter C." Then put the bits of paper in a hat.
2. In small groups, pick two questions out of the hat and take three to five pictures of things around the room that fit each condition.
3. Create a photo album using a presentation app like Keynote, creating a section for each condition. Don't label the conditions yet.
4. Present your photo album to another group to see if they can guess the conditions. If there's a correct guess, add the conditional statement—for example, "If blue, then take picture"—to the album page.

Think about it: Were there cases that were difficult to judge whether a photo matched? How would a computer handle these cases?

MeeBot Dances (1 session)

1. If you have a MeeBot robot from UBTECH, download the MeeBot Dances lesson in Swift Playgrounds.



2. Go through the lesson as see how you can apply functions, for loops, and conditional code to make MeeBot dance.
3. Create your own dance routine. You can even choose your own music on the last page.

Think about it: How did you use code to reflect the tempo of the music?

Think Again and Again: While Loops

4

Introduction (5 minutes)

In topic 2, you learned about functions and for loops. What was the dance you did together? How would you use for loops to write the function for the dance?

Now think about if you wanted to do that dance at the school dance. How would you know when to stop dancing when the song ended?

You'd use a conditional code—if song plays, then dance. Or more clearly, a while loop—while the song plays, dance.

This is different from a for loop, which tells a computer to run a block of code a certain number of times—for example, dance in a circle 10 times. A while loops tells the computer to run a block of code until something happens. So dance in a circle until the song stops playing.

Watch this video to learn more about while loops.

Practice (40 minutes)

Now use Swift Playgrounds Learn to Code 1 to complete the puzzles with green checkmarks in the list at right.

Think about it: When did you use for loops and while loops? How did you decide?

While loop: A loop that runs a block of code as long as a given condition is true. When the condition is false, the loop stops running.

While Loops

- Introduction ✓
- Running Code While... ✓
- Creating Smarter While Loops ✓
- Choosing the Correct Tool ✓
- Four by Four
- Turned Around
- Land of Bounty
- Nesting Loops ✓
- Random Rectangles
- You're Always Right

Think Again and Again: While Loops
Take It Further

4

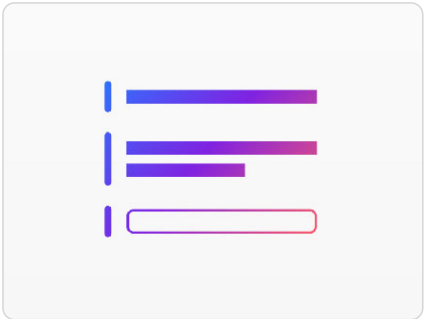
Hide and seek again (1 session)

1. Hide a small object in or near the room.
2. Stand in one spot and use the iPad camera to record yourself giving directions for someone to find the object. The directions should start from where you're standing. Use functions, for loops, and while loops where you can.
3. Now trade videos with another student. Watch their video to try to find the hidden object. Did you find it?
4. Review both videos together. Write down the instances where you used for and while loops in the directions.

Think about it: Did using loops make things easier this time? Were there cases where it was harder?

21 Questions (1 session)

1. Let's take a look at the Answers Starting Point again. Download this version of it.



2. This Answers template is set up for you to play 21 Questions. Take a look at the code first. What will the code do?
3. The creator decides on an object or a thing that they enter as the answer in the code. The player can ask the creator 21 yes or no questions.
4. After each question, the player can enter their guess in the live view to check it. Be sure the creator hands over their playground with the live view in full screen so the player can't see the code and the correct answer.

Think about it: What coding concepts did this playground use, and how are they being used?

Think the Same Idea: Algorithms

5

Introduction (5 minutes)

As a group, name some things you do all the time that require multiple steps to complete—for example, brushing your teeth or making a sandwich. These are all algorithms.

Pick one example and have multiple students give directions for how they do it. Were their directions the same? Where did they differ? Did they all accomplish the same thing in the end?

Watch this video to learn more about algorithms.

Practice (40 minutes)

Now use Swift Playgrounds Learn to Code 1 to complete the puzzles with green checkmarks in the list at right.

Think about it: How many different ways do you think there are to solve each puzzle? Who had the shortest algorithm? Who had the most interesting one?

Algorithm: A step-by-step set of rules or instructions.

Pseudocode: An informal description of code or a concept that's intended for human reading.

Algorithms

- Introduction ✓
- The Right-Hand Rule ✓
- Adjusting Your Algorithm ✓
- Conquering a Maze ✓
- Which Way to Turn?
- Roll Right, Roll Left

Think the Same Idea: Algorithms
Take It Further

5

Who's the tallest? (1 session)

1. Split the group into a few small groups. Each group will come up with a way, or an algorithm, for someone to determine who the tallest person is. It doesn't count if you just know or can tell just by looking!
2. Draw on your coding knowledge to come up with the steps for your algorithm. You can invent your own pseudocode to write your algorithms with as much coding terminology as possible.
3. Have each group present their algorithm. The entire group should perform it.

Think about it: Which algorithm seems most efficient? If you wanted to find the shortest student, what would you change in your algorithm?

Parrot (1 session)

1. If you have a drone from Parrot available, download the Parrot lesson in Swift Playgrounds.

2. Use all the coding skills you've learned so far to make the drone take off, land, move in all directions, and make acrobatic figures.
3. Finally, create an algorithm for your drone's flight path to get from point A to point B.

Think about it: How did the drone's speed affect your flight paths?

Think Like a NewsBot: Variables

6

Introduction (5 minutes)

Imagine you're moving to a new home. You'd probably put your belongings into boxes, labeling them with a word or two to describe the contents. For example, if a box contains trophies, you might label the box "Trophies." This example suggests three important features of a container.

When we program a computer, we use something called variables instead of boxes. Variables are similar to boxes—they have both a label (a name) and contents (a value). The value or contents can change, but the label or name can't. We can find a variable's value or contents by finding the variable with the correct name or label.

Watch this video to learn more.

Practice (40 minutes)

Now use Swift Playgrounds Learn to Code 2 to complete the puzzles with the green checkmarks in the list at right.

Think about it: How did using variables help you with the app?

Variable: A named container that stores a value. The value can change over time.

Variable

- Introduction ✓
- Keeping Track ✓
- Bump Up the Value
- Incrementing the Value ✓
- Seeking Seven Gems ✓
- Three Gems, Four Switches
- Checking for Equal Values ✓
- Round Up the Switches
- Collect the Total

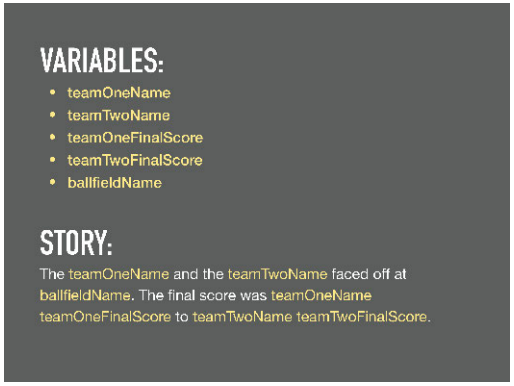
Think Like a NewsBot: Variables
Take It Further

6

Newsbot (1 session)

1. For this activity, you'll create NewsBot—a robot that can automatically write a short article. First, think about a news or sporting event, and what pieces of information you might need to write about it.
2. Brainstorm four to six variables, like a team name, score, and event date. Write a two- to three-sentence story using the variables, which NewsBot can then fill in for other similar events.
3. Pair up. One of you provides the information for each variable to fill in the story. Then switch. Do you have two complete stories that make sense?

Think about it: Did any variable names work better than others? Why or why not?



Sphero Arcade (1 session)

1. If you have a Sphero SPRK+ robot available, download the Sphero Arcade lesson in Swift Playgrounds.

2. Apply functions and variables to aim, detect collisions, and eventually build your own version of Pong.
3. On the Play Sphero Pong page, analyze the code before running or editing it. What does each command do?
4. What elements of the game did you change using variables?

Think about it: What other games could you customize in the way that you did with Pong?

Think Like an Architect: Types

7

Introduction (5 minutes)

How many different types of buildings can you think of? Pick one type. What makes it unique? In other words, what are some specific features of that particular type of building? What typically happens in it? We'll call these behaviors. For example, a school has classrooms (features) and the bell rings between class periods (behavior). Does everyone agree on the properties and behaviors? Why or why not?

If we use a computer program to help us construct the building, we have to be very specific. We need to define its type by providing the properties (which we called *features*) and methods (what we called *behaviors*).

Watch this video to learn more.

Practice (40 minutes)

Now use Swift Playgrounds Learn to Code 2 to complete the puzzles with the green checkmarks in the list at right.

Think about it: What were the types in the app? What did you initialize?

Type: A named grouping of properties (the features) and methods (the behaviors) of a kind of data.

Initialization: The act of creating a new instance of a type, which includes setting initial values for any properties of the type.

Types

- Introduction ✓
- Deactivating a Portal ✓
- Portal On and Off
- Setting the Right Portal ✓
- Corners of the World
- Random Gems Everywhere

Initialization

- Introduction ✓
- Initializing Your Expert ✓
- Train Your Expert
- Using Instances of Different Types ✓
- It Takes Two

Think Like an Architect: Types
Take It Further

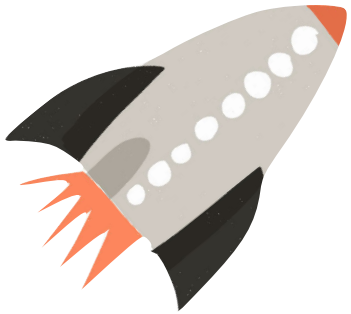
7

Be an architect (1 session)

1. Choose a type of building, either real or imagined.
2. Think of five to six variables to describe what your type of building looks like. These variables shouldn't contain any values; instead, they should be descriptive of values—like "height" or "numberOfWindows."
3. Add values next to the variables. This describes a specific instance of the building type. In Swift terminology, you're initializing an instance of the building type.
4. Finish the initialization by using a drawing app like Notes to sketch the building type, reflecting the variables and values.
5. Find a partner and trade the lists of your building types' variables and values. Draw each other's building types, then share your drawings. How similar are they?

Think about it: Can you think of ways to make the drawings look more alike?

Type: rocketHouse
numberOfEngines = 4
height = 15 meters
numberOfWindows = 8
color = Space Gray
doorShape = Rounded rectangle



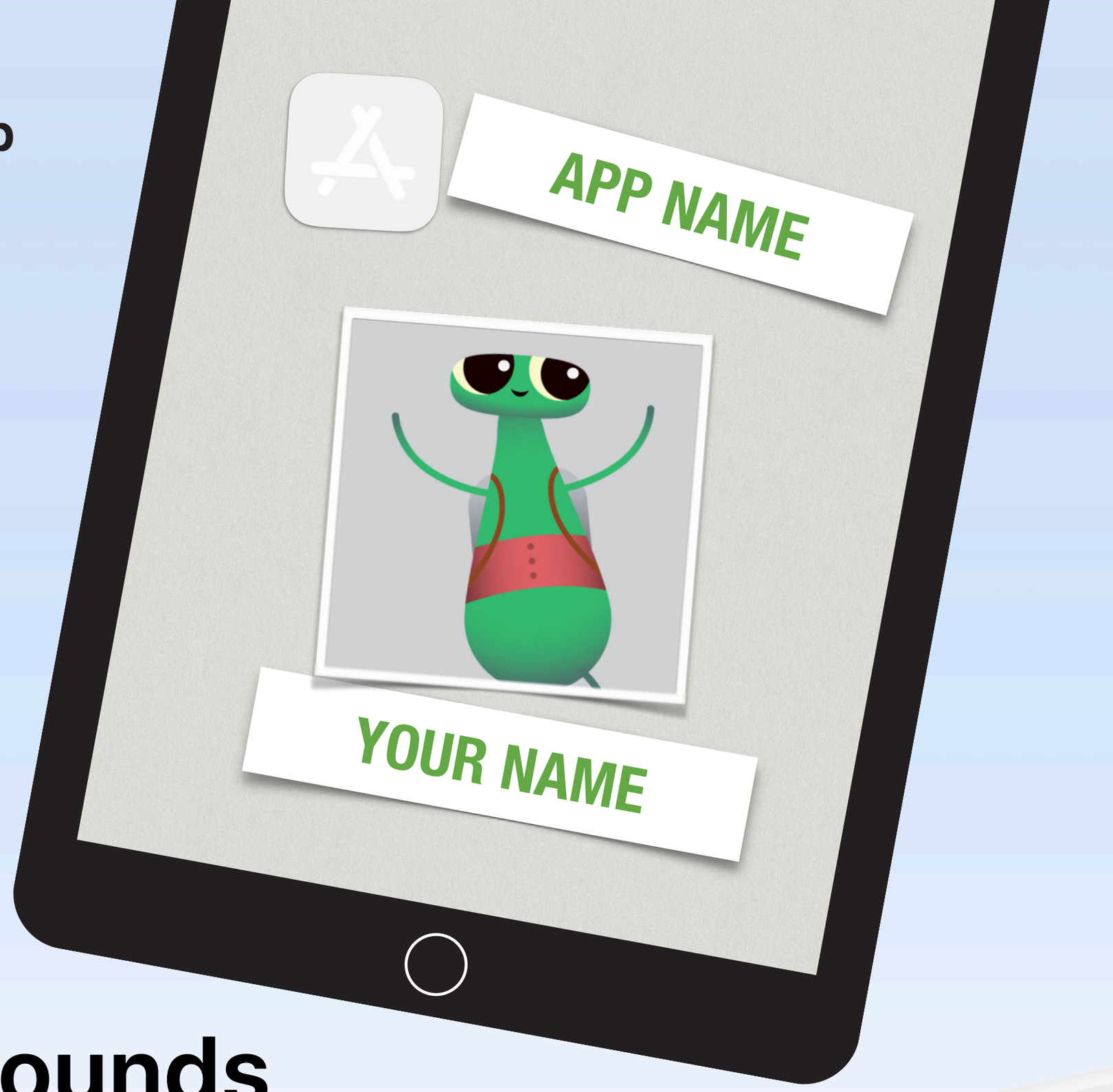
Rock, Paper, Scissors (1 session)

1. Download the Rock, Paper, Scissors challenge in Swift Playgrounds.

2. Explore the lesson. How does the code work? What coding concepts do you recognize?
3. Now try to create a version of the game in which you use your own rules and add new actions. To do this, you'll need to define the game's type and initialize it, along with the associated properties and methods.

Think about it: What were the properties and methods in the original game? What were the new properties and methods you added to personalize the game?

Swift Coding Club



Swift Playgrounds

App Design Journal

Welcome

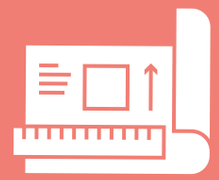
Your Keynote app design journal will help you keep track of your ideas and guide you as you cycle through the four stages of the app design process. You can play it through to see what's in it, but you'll work in slide view to add notes, images, shapes, and more. Feel free to add and duplicate slides, and refer back to it during current and future app projects.

Brainstorm

Purpose
Ideas
Audience
Focus
Reiterate

Plan

UI/UX
iOS features
Design



Prototype

Design
Flowchart
Build

Evaluate

Observation
Interview

Brainstorm

Purpose
Ideas
Audience
Focus
Reiterate

Overview

The brainstorming stage allows you to identify problems and come up with possible solutions. This section includes a few key topics for you to think through. Some topics have optional Go Further activities if you're interested in exploring more. Jot down as many ideas, notes, and sketches that can help you design an app for solving a problem in your community.

Brainstorm

Purpose
Ideas
Audience
Focus
Reiterate

Define the opportunity, problem, or challenge

Before you can start to explore different options for your app, you need to be clear on what the opportunity, problem, or challenge is.

1. What do you know about the opportunity, problem, or challenge?

2. What questions do you need to find answers to?

3. Can you explain the opportunity, problem, or challenge in just one sentence?

Brainstorm

Purpose
Ideas
Audience
Focus
Reiterate

My favorite apps

Think about the apps that you use. Identify each app's purpose and why you use it. Which do you use most, and which did you stop using after just a few times? Why did you download them in the first place? Brainstorm a list of your favorite apps, and identify their purposes and the features that make them good.

	App purpose:	I like this app because . . .
	App purpose:	I like this app because . . .
	App purpose:	I like this app because . . .
	App purpose:	I like this app because . . .
	App purpose:	I like this app because . . .

Brainstorm

Purpose
Ideas
Audience
Focus
Reiterate

My ideas

Brainstorm a list of apps you'd like to build. These could be new ideas, apps to solve specific problems, apps you think you can improve or personalize, or something silly! Browse the App Store for inspiration. Keep adding to this list and revisit it, as some ideas might become more or less interesting or crazy in the future.

Add your ideas.

Brainstorm

Purpose
Ideas
Audience
Focus
Reiterate

My app idea

From your brainstorming list, select one app idea to develop further and describe it below.

App name:

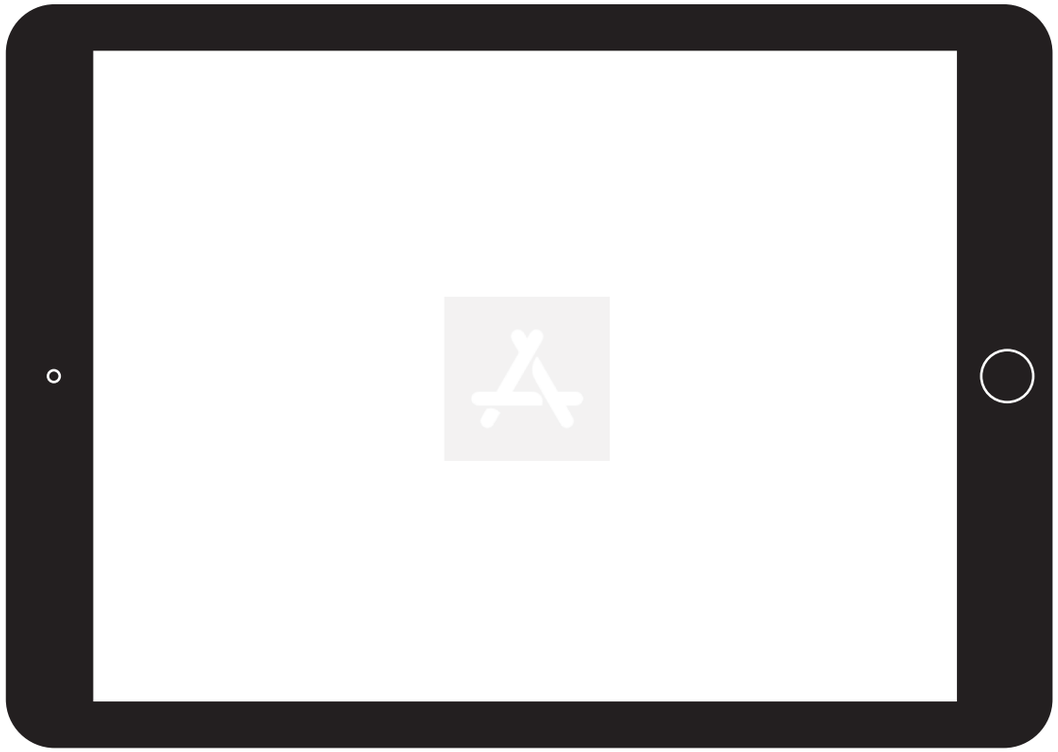
Write a description of what the app does.

Go Further
Do a little research on your initial top ideas. Take a good look at the App Store. Do your app ideas already exist? Don't be discouraged if you find one or several that are like the app you've imagined. It just means you had a good idea. And maybe now that you've seen a bunch of similar apps, you can see ways of improving them.

For your favorite app idea, identify its top competitor on the App Store. If you can, download the app. Then check it out to answer the questions at right.

If there are any user reviews on the App Store, be sure to read those carefully, too. What are people finding difficult about the app? What else do they wish the app could do? Where are people getting confused? How would your app address each of these concerns?

My app's top competitor



Add example images of your app's top competitor here.

Is it easy to use? Why or why not?

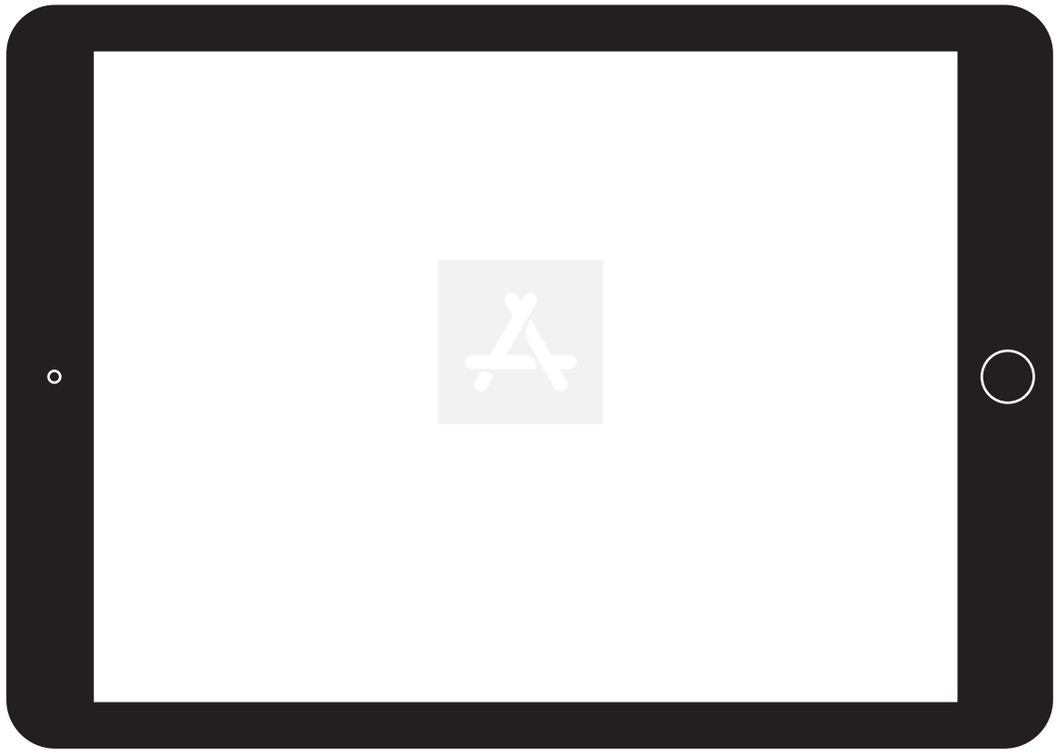
Can you suggest improvements to the user interface?

How could it be designed better?

Brainstorm

Purpose
Ideas
Audience
Focus
Reiterate

It’s important to design with a target audience in mind. Who do you want using your app?



Explore apps from different categories in the App Store. Take a look at the icons, screenshots, and descriptions. Add example images here to keep track of your research.

What have you learned about the audience those apps are meant for?

Did the developers do a good job communicating that?

Judging from the screenshots or preview video, do you think the apps are appropriate for their intended audiences?

Go Further

For your app idea, create a persona for each type of person who would use the app.

Duplicate this slide to outline each persona.

What does this person do?

How old is the person?

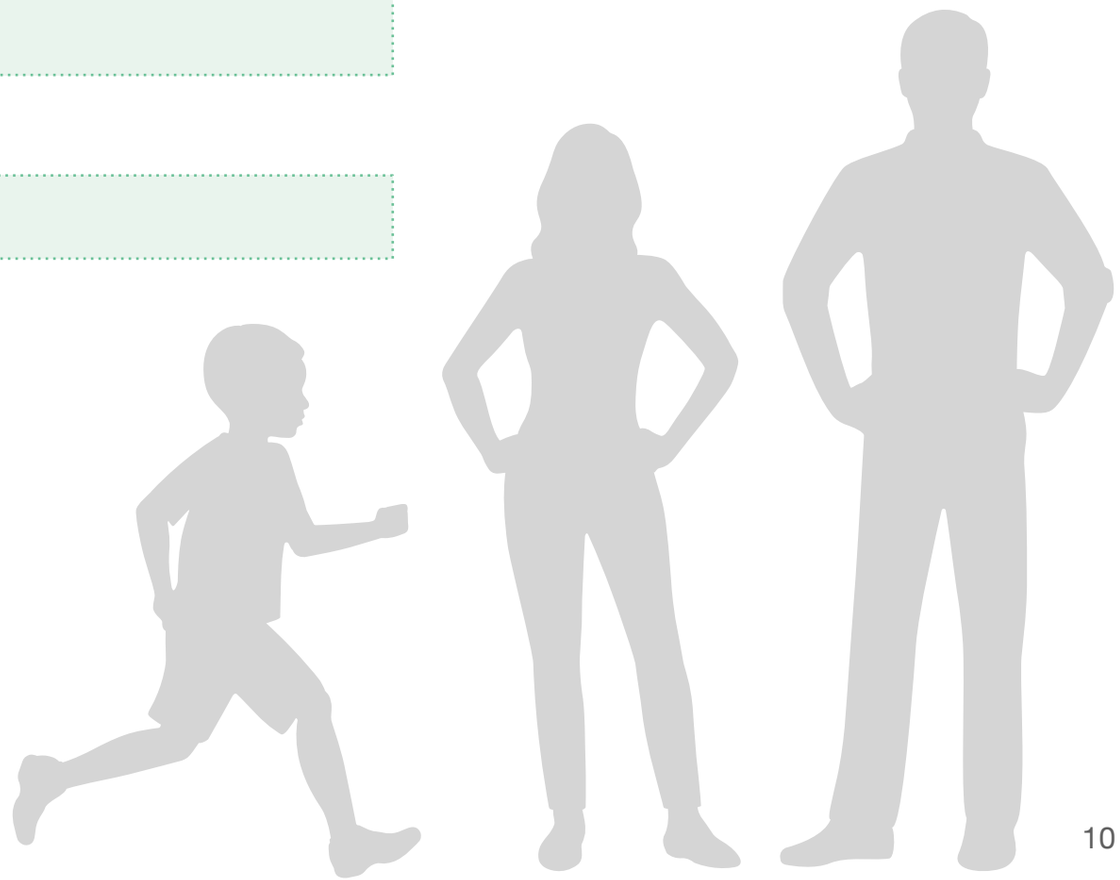
Why is the person using the app?

Does the person prefer pictures or words?

How often does the person use their device?

Include other details.

[Optional] Illustration or stock photo of the persona



Brainstorm

Purpose
Ideas
Audience
Focus
Reiterate

Focus

Before you commit to your app, go back and review your list of app ideas. Which ones seem most interesting? Focus on a few ideas for further brainstorming. What purposes do they serve and how do they solve issues? Who are the audiences? Write app statements to clearly define the apps' purposes. This can help you decide whether they're good ideas or not. Compare your new ideas to your favorite app idea. Is it still your favorite?

What will your app do?

My app will . . .

Why does this need exist?

because . . .

What will your app do?

My app will . . .

Why does this need exist?

because . . .

What will your app do?

My app will . . .

Why does this need exist?

because . . .

Plan

UI/UX
iOS features
Design

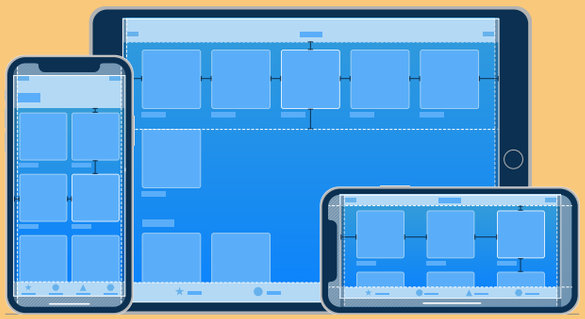
Overview

The planning stage is when you figure out the details of your app and how it can achieve its goal.

Consider these three key areas as you develop your apps: UI/UX, iOS features, and design. The more you learn about each topic, the more advanced your app designs will be. The Apple Developer website is a good resource for you to learn more about these topics.

Plan

UI/UX
iOS features
Design



UI/UX

A good app should be easy to use. That's where the user interface (UI) design comes in. A well-designed UI makes for a good user experience (UX). Think back to the first times you used some of your apps, or try using a new app. What was the experience like? Did you get confused navigating them? Review elements like font size, icon shape and placement, and the navigation from screen to screen. Even the smallest element makes a difference in how someone experiences your app. Be sure to review the Human Interface Guidelines.

Go back to your list of favorite apps and choose one to review. Think about the features that make it easy to use.

Review of my favorite app

Go Further

Now consider the rest of the list of your favorite apps. Rank them in terms of their UI design. Which apps are easy to use and seem to just work? Write down the reasons that some apps are easier to use than others. Did you know what to do immediately? How many taps did it take to get going on the app? The answer should be very few. First impressions count. Compare your notes with other students. Did you agree on the reasons?

1.

2.

3.

4.

5.

The basics

Keyboard
It's a very basic feature, but the keyboard is essential to many apps so users can input names, numeric data, and even emoji. It's necessary for many apps, including email, a word cloud app, or a translation app where you type a phrase and hear it in another language. What apps can you think of that use the keyboard? What kind of data do the apps take in?

Camera and microphone
Many apps use the camera and microphone to record sights and sounds. Think about apps that let you create movies, music, and photo albums. What about apps that let you communicate, like FaceTime and Messages? Or analysis apps, where you can overlay a graph onto a photo or mark it up for analysis? How many ways can you think of to use the camera and microphone?

Touchscreen
Both iPhone and iPad have a touch-sensitive screen. You can create apps that detect a user interaction, such as tapping the screen once, double-tapping, swiping, or dragging a button or an object. Think of the possibilities for games and other user interfaces that use touch as a very natural interaction with elements on the screen.

How might you use these features in your app?

Get connected

Wi-Fi
Does your app need to connect to the Internet to work? While most people may have access to Wi-Fi, think about what happens with your app when a user can't connect to a Wi-Fi network. Does having Wi-Fi access fit with your target audience persona?

GPS
iOS devices have a built-in GPS (global positioning system) that shows where they're located on the earth within about 15 feet. It can also detect altitude (vertical distance from sea level). The Maps and Weather apps on your iPhone use GPS.

Bluetooth
This technology lets iOS devices connect with other nearby devices, such as a speaker to play music, a robot like Sphero that your device can control, or a digital thermometer.

How might you use these features in your app?

Get innovative

Speech recognition and machine learning
Try Siri out. Siri can recognize your speech. And if you continue to use Siri, you'll notice that it gets better at knowing what you want. That's the machine learning part. Which apps that typically use the keyboard to collect information can instead use speech recognition and machine learning? What types of audiences would benefit from these features?

Accelerometer and gyroscope
The accelerometer can detect whether a device is accelerating, decelerating, or in zero gravity. The gyroscope references direction, so it can measure the rotation of a device. Together, they can detect how iOS devices are being moved in three-dimensional space. Could you create an app that recognizes if the user is falling? Think about the Health app and the level tool in the Compass app on iPhone. How do they use the accelerometer and gyroscope?

Augmented reality
With augmented reality, you can blend digital objects and information with your real-world environment. Imagine seeing a life-size elephant right in your backyard or seeing the images in your favorite book come to life.

How might you use these features in your app?

Accessibility

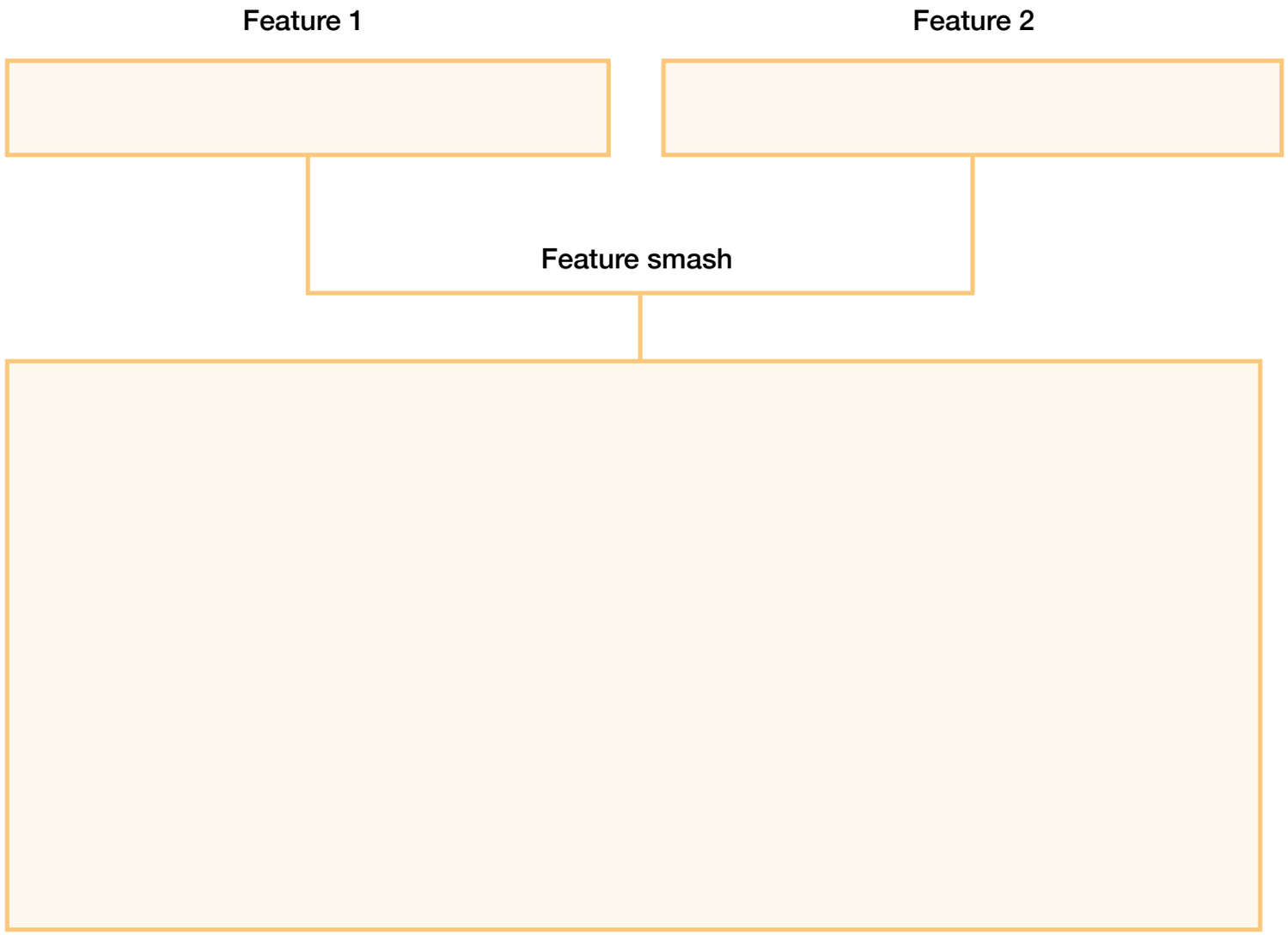
iOS supports multiple ways to access onscreen content. People with physical impairments can use Siri or Dictation to interact with apps. People with visual impairments can increase the size and contrast of screen elements, or use the VoiceOver screen reader and navigate your app entirely by sound. But iOS accessibility features don't help only users with disabilities—they can help all users access your app in whatever ways they're comfortable with.

Try the iOS accessibility features so you know firsthand how they work.

How might you use these features in your app?

Feature smash

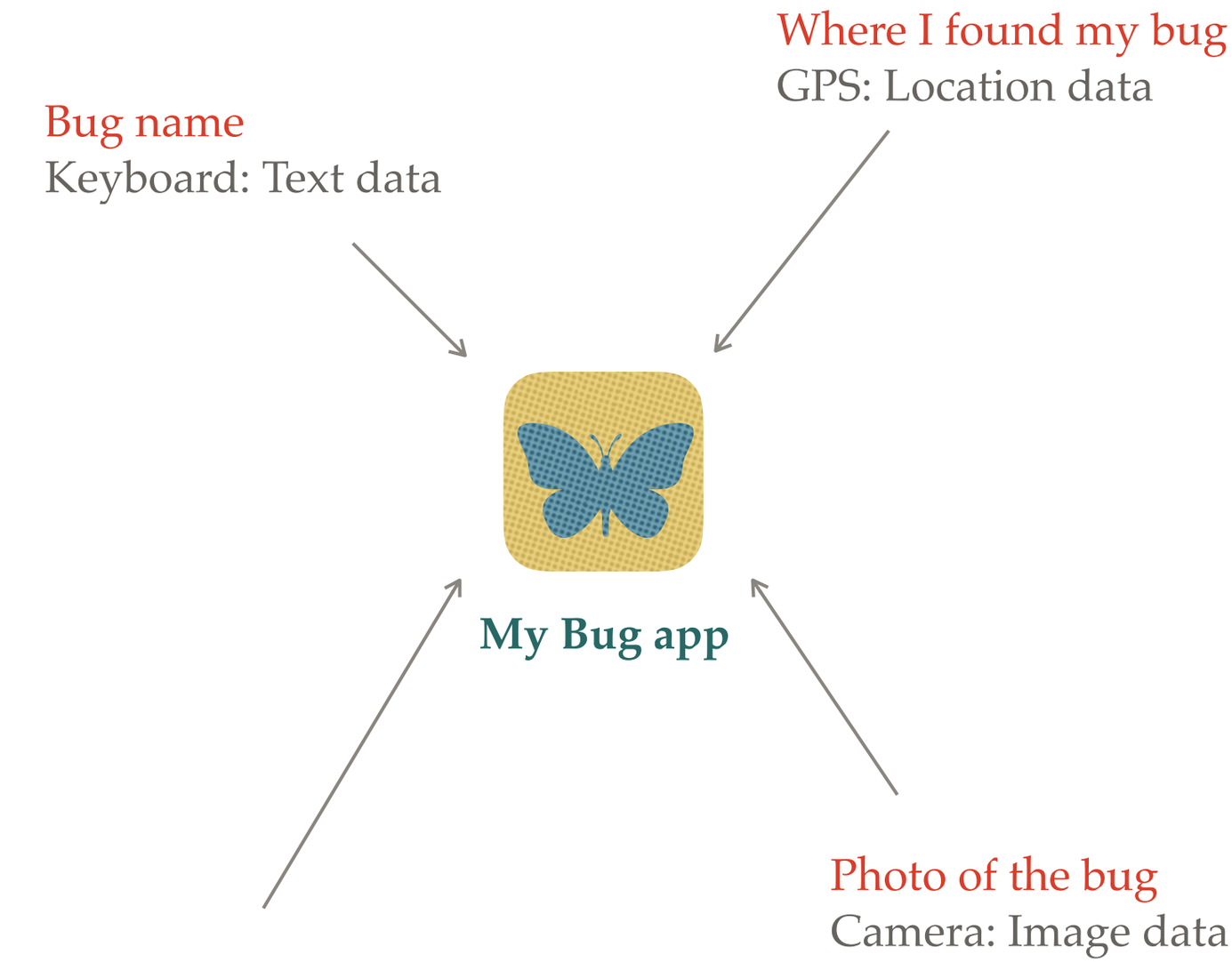
Now that you've learned about various features, it's time to see what combination works best for your app. First, stretch your imagination and try combining different features. Write down all the features on separate pieces of paper and fold them up. Take turns drawing at least two pieces of paper and coming up with ways to use those features together. For example, you could use the accelerometer and Bluetooth together to connect to a robot and use your device as the remote control.



Go Further
Now think about your app. Which features are essential to make your app state of the art? Which ones give it a wow factor? It could be, “Wow, that was so easy!” or “Wow, I’ve never seen that before!” Remember, you want your app to be unique, but also simple and easy to use.

Insert a new slide and map out what features you want to include in your app, and how they help your app accomplish its goal.

Example:



Design

Give your app some style and personality! But remember to keep it simple. You want the purpose of your app to shine through; you don't want too many colors or unnecessary elements to distract your users. Create a mood board to guide your app design.

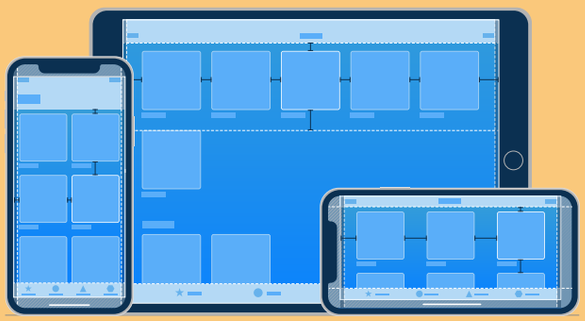
Choose a color scheme.

Add example images of the types of visuals needed.

What fonts will your app use?

Add phrases of design principles, such as "Keep it simple," "Modern," and so on.

Add sound files or describe the sounds your app will use to notify users of something, immerse them in a game atmosphere, or enhance the app mood.

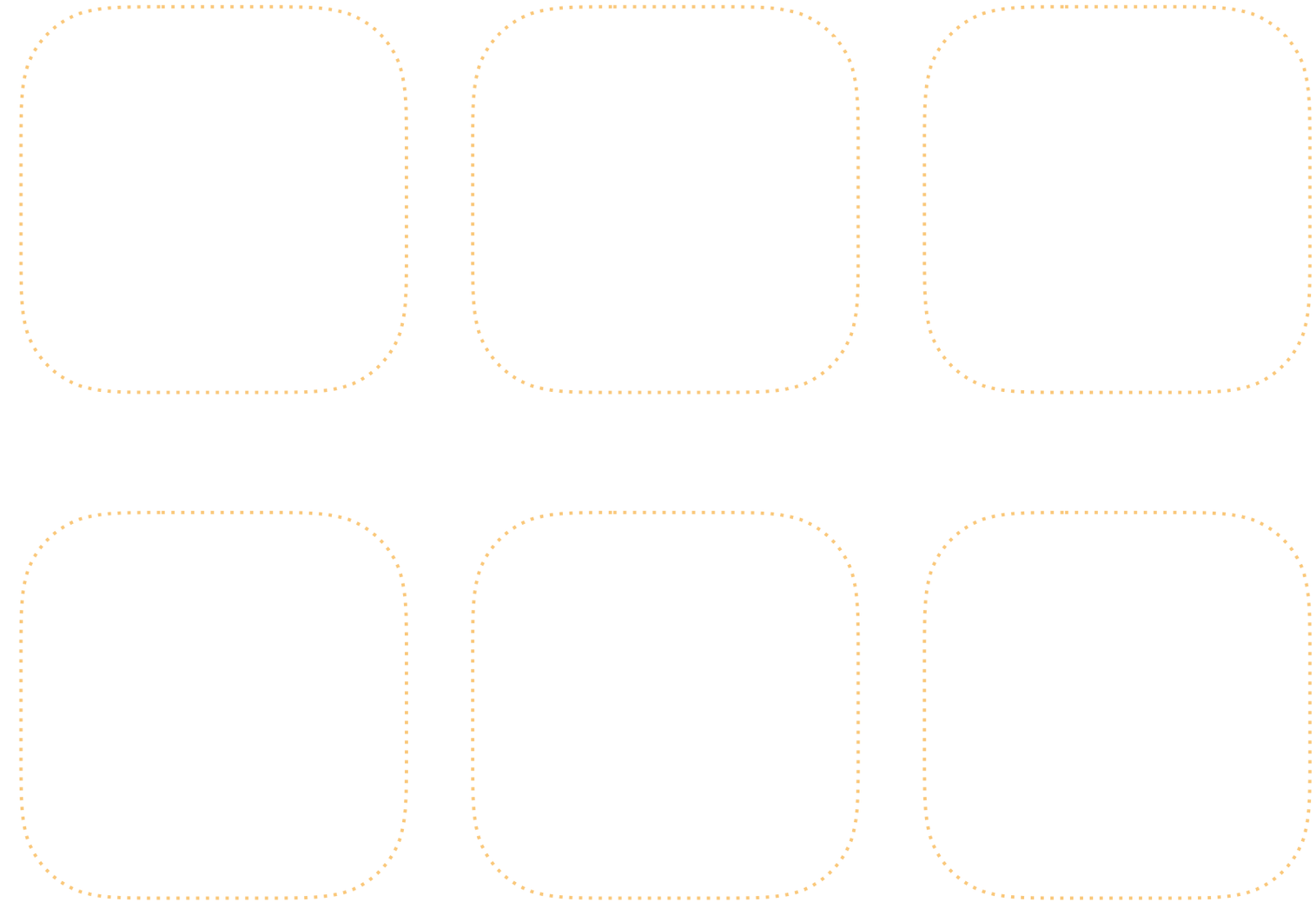


Be sure to review the Human Interface Guidelines again.

Go Further
An appealing icon matters to give a good first impression. When people are searching the App Store, they notice a good icon. You might have a brilliantly coded app with a slick user interface, but people will never download it if your app icon doesn't convey the right message.

Design a few different icons for your app. Show them to other students and ask them what they think your app does based on just the icon. Which one did they like the best? Pro tip: You can take a screenshot of your home screen and add in your icon to see if it stands out.

Design a few different icons for your app



Prototype

Design
Storyboard
Build

Overview

Take a look at this video from WWDC on 60-second prototyping to find out how you can use Keynote to quickly test ideas. For a more in-depth exploration of app prototyping, watch this video from WWDC on iterative design. You don't need to watch the whole video, but it should give you a better idea of what to expect when prototyping with Keynote.

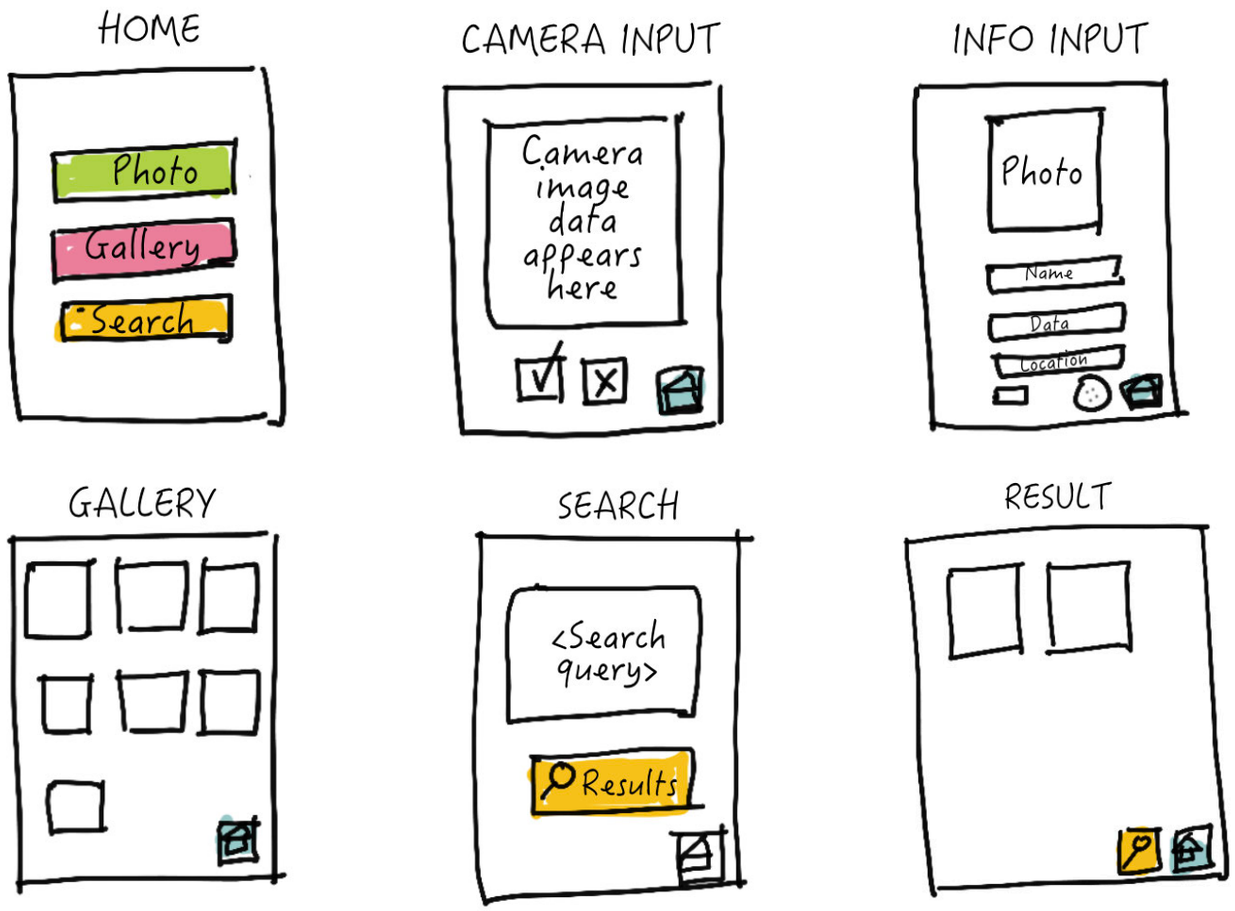
Prototype

Design
Storyboard
Build

Design

What do the main screens of your app look like? What features appear where? Take a look at your design mood board again and mock up a few screens. You can use Keynote or a drawing app, or draw on index cards to create each screen. Take screenshots or photos of your images and add them to the app you'll build your prototype in.

Tip: Not feeling artsy? Use existing apps as inspiration. Take screenshots of app screens that have great features, add them as a layer or template, and customize them.

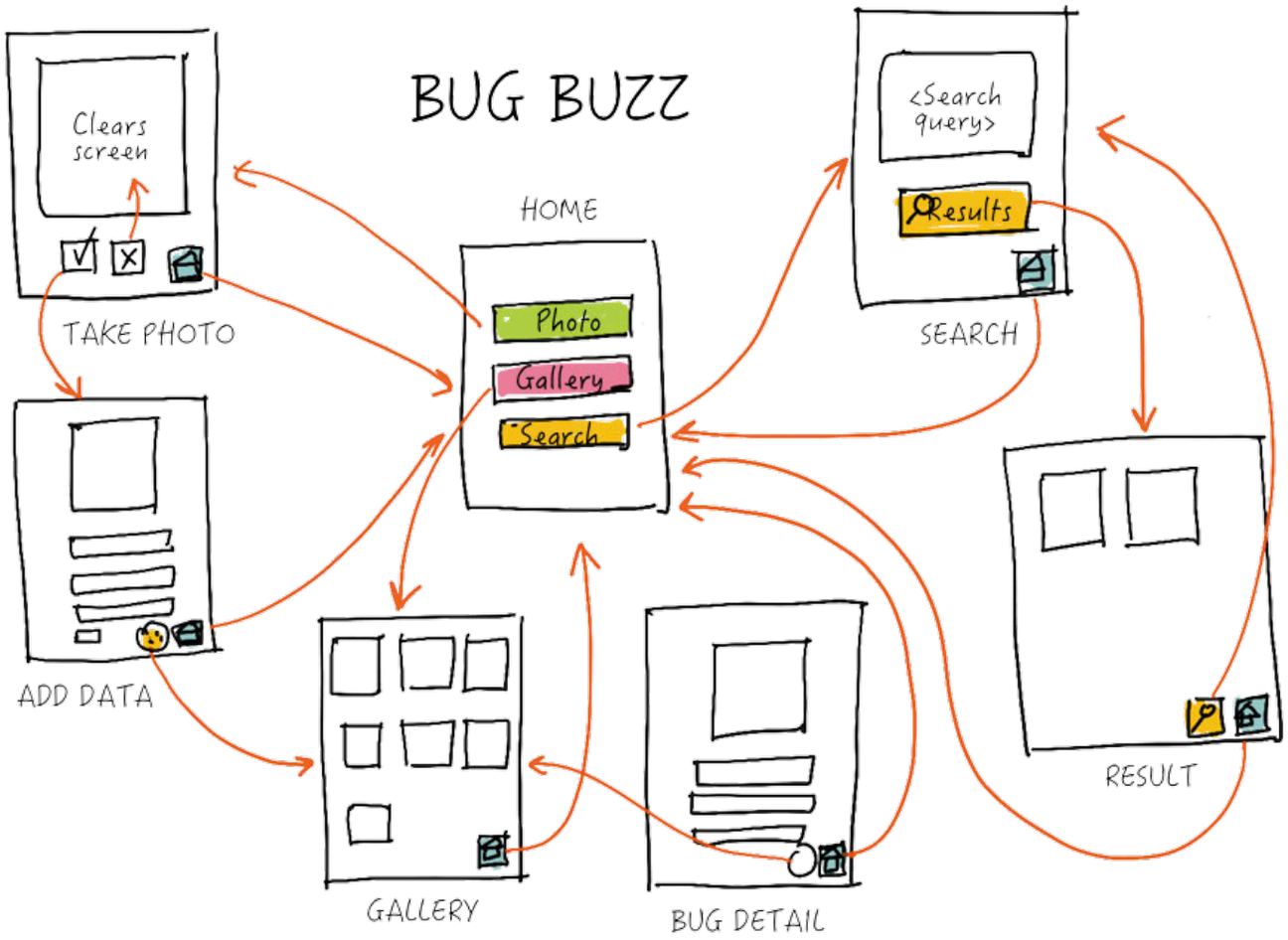


Prototype

Design
Storyboard
Build

Storyboard

Insert a new slide, and create a flowchart like the sample below using the images you created. What are the key stages of the app, and how does the user get there? Be specific. For example, at what point is a certain feature available? Or, what happens when a user taps yes or no on the screen? What data does your app collect or provide? How is it represented? Map out the conditional statements necessary for your app—for example, if user taps yes, then x; else, y. What other coding concepts would apply in your app? Are there parts that repeat and would use a loop?



Prototype

Design
Storyboard
Build

Prototyping in Keynote

1. Set up the Keynote document to be the right size for your app prototype to run on your demo device. Tap the More button (…), then tap Document Setup. Tap Slide Size, choose Custom Slide Size, then enter one of these sets of dimensions:

- iPad: width=834 pts, height=1112 pts
- iPhone 8: width=375 pts, height=812 pts

2. Decide on the colors and fonts you'll use, then design the navigation buttons. Park these design tools in working slides that you can delete later.
3. Build each screen on a different slide.
4. Create interactive links between the slides so that their buttons trigger touch events. Tap the object you want to link, tap Link, then choose Link To Slide.
5. To make sure that the presentation changes slides only when the user taps the navigation buttons, tap the More button (…), tap Presentation Type, then tap Links Only.

Build

First, take a look at this finished prototype file for inspiration. Then use Keynote on a Mac or iPad or use an iOS app like POP - Prototyping on Paper to build your prototype.

For each version of your prototype, think about the following:

- Can users choose to engage with the content in different ways?
- Can you provide different representations of the same data?
- What's the first screen (view) that the user sees? Which buttons are visible? Then what happens?
- Decide where and what kinds of graphics and icons your app will display.
- How many taps will it take for users to find out what they need to know?
- How would users navigate between views?
- What are some simple ways to communicate the features of your app without using words?

Evaluate

Observation
Interview

Overview

Now it’s time to test your prototype. You can have your classmates, family, and others try it. If possible, try to find testers who fit the target audience for your app. You should present your prototype, explain your new app idea, and tell the testers that you want them to try it. You can provide guidance if needed, but the objective is to observe the user and ask questions later.

Evaluate

Observation
Interview

Observation

Watch the tester explore your app, and use the questions below as guidelines for recording your observations.

Did the user know what buttons to tap?

Was the user ever confused? At what point?

Did the user enjoy the app?

Did the user smile or laugh at specific points?

Did you observe anything else?

Evaluate

Observation
Interview

Interview

After the user finishes testing your app, interview them to better understand their experience. Here are a few questions to get you started:

- What did you like and not like about the app?

- Is the app useful? Would you use an app like this?

- What more might you want to see in this app?

Reiterate

Remember, this is a design cycle and it's time to go back to the brainstorming stage. As you repeat the design cycle, think about what you learned from your evaluation. Did problems come up, and if so, how can you fix them? How can you improve your app?

Another important question to ask yourself is whether you're still excited about your app idea. If not, it might be time to go back to your list. Not all ideas pan out. One objective of the design cycle is to help you test concepts and figure out what's worth pursuing.

Do you still want to continue with your idea? If so, write the name of your app below, score it out of five stars, and write an app review.

Review of my app

Go Further
Revisit your criteria for what makes an app great from the Purpose topic and answer the questions on the right.

Continue to revisit the different topics throughout the design cycle. Revise your prototype accordingly, testing and retesting until you have the next great app.

Is your app innovative?

Does it do something that existing apps don't do?

Is it an app someone would use over and over again?

How can you improve your app?

App Pitch

You've tested and improved your app idea. Now it's time to polish it up and share it! Make a three-minute presentation or video of your pitch. A good pitch will tell a strong and clear story that makes people want your app!

Your pitch should include:

- **Why:** The problem your app is trying to solve
- **Who:** A description of who your app is for
- **What:** An overview of the app
- **How:** Details about the UX and UI, including:
  - The design
  - The features
  - The coding concepts it uses
  - The prototype and any visuals
  - Improvements made based on user testing