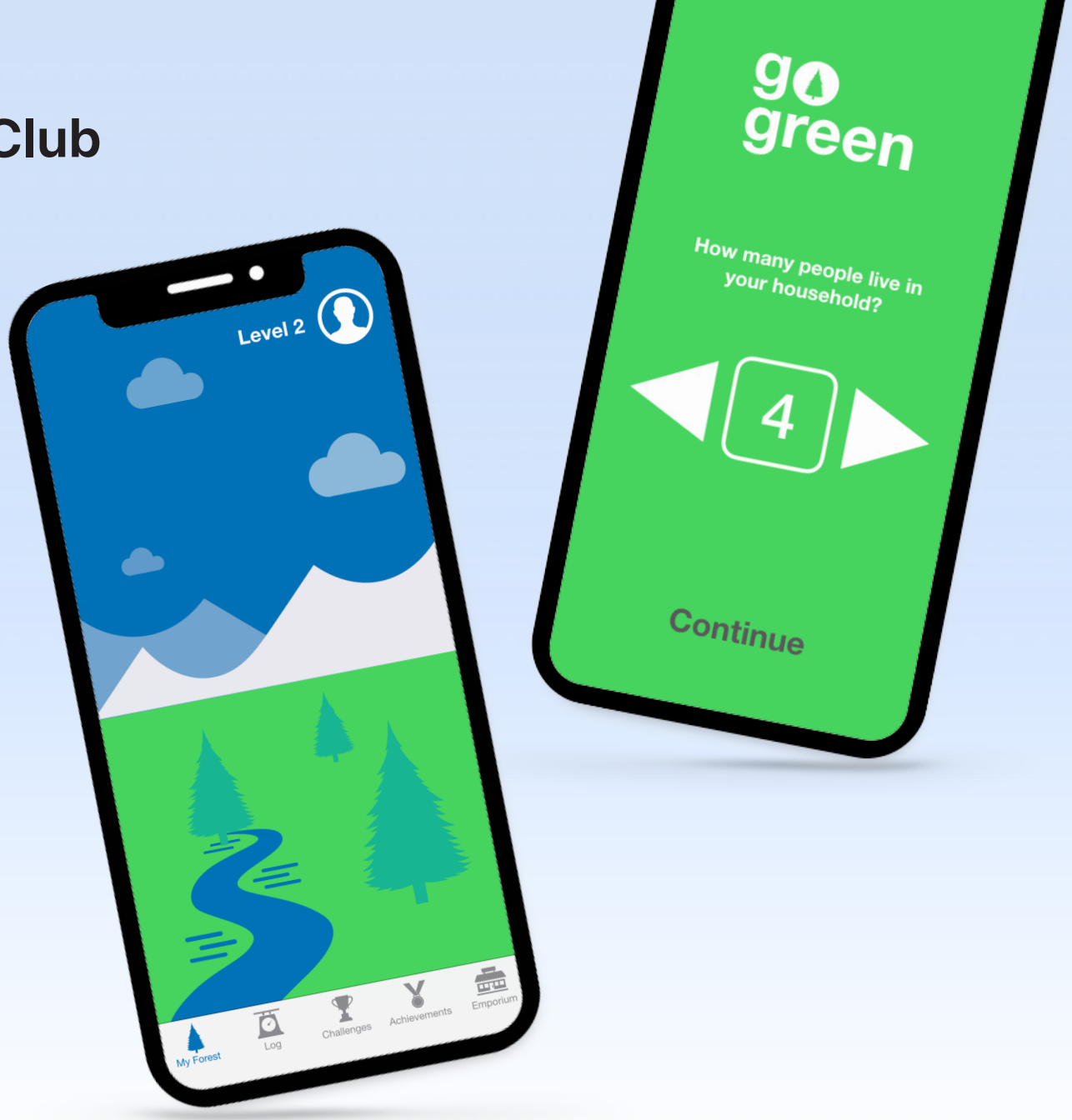
Swift Coding Club



Xcode 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.

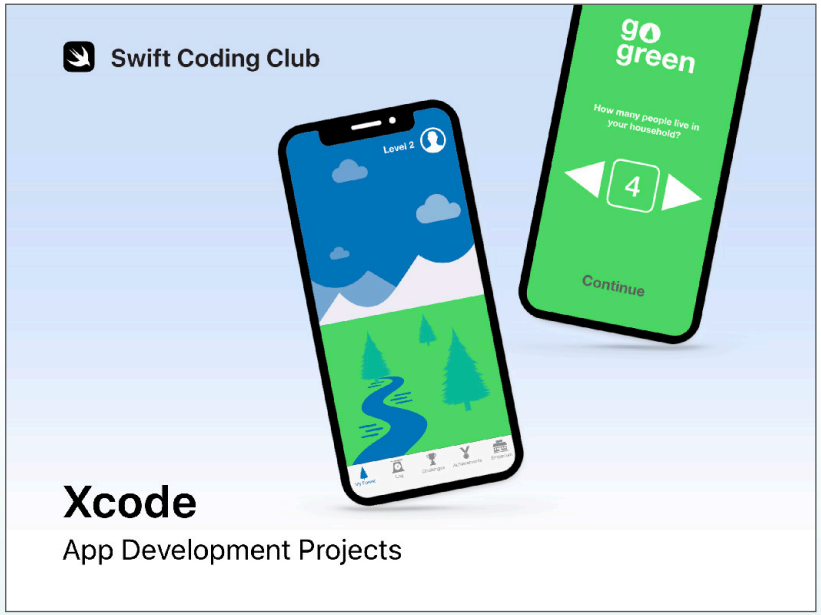
Xcode | Ages 14+

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.



App Development Projects
Guided app development projects introduce the same tools, techniques, and concepts that professionals use. Learn programming concepts as you write in Xcode playgrounds on Mac.

Download Xcode App Development Projects >

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

Download Xcode App Design Journal >

2. Check your tech.

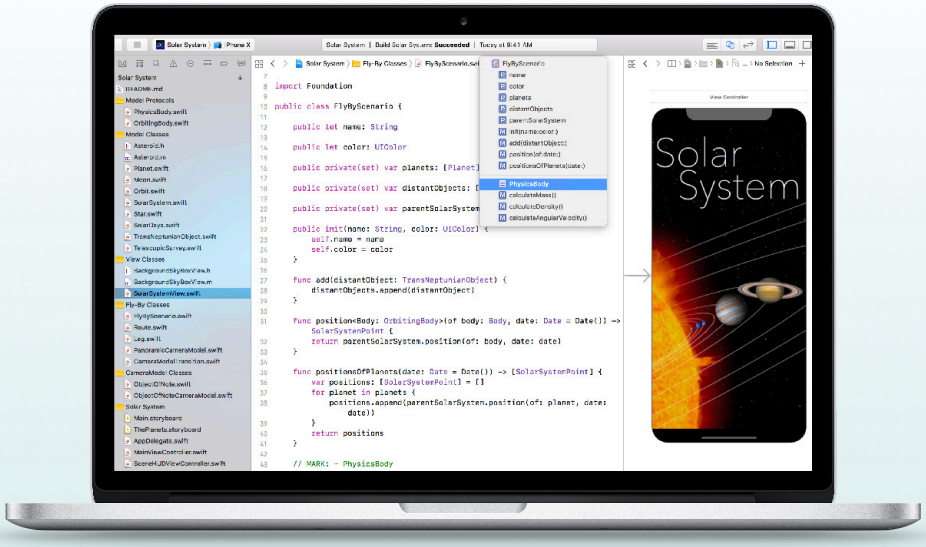
Before your first meeting, be sure you have the following:

- **Mac.** Students will need Mac computers running macOS Mojave or High Sierra. It's recommended that they each have their own, but they can also share and code together.
- **Xcode.** This free Mac app from Apple is used to build every other Mac app and every iOS app, too. It has all the tools for creating an amazing app experience. Xcode 9 is compatible with Swift 4, and Xcode 10 is compatible with Swift 4.2.
- **Intro to App Development with Swift.** This free resource from Apple guides beginners through eight coding projects.
- **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?



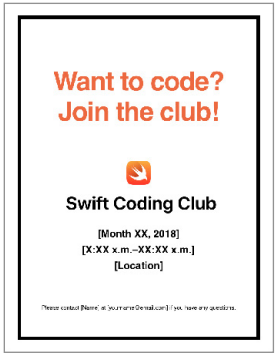
4. Spread the word.

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 projects. 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
- App Development Project 1
- App Design Journal: Brainstorm

Sessions 6–10

- App Development Project 2
- App Design Journal: Plan

Sessions 11–15

- App Development Project 3
- App Design Journal: Prototype

Sessions 16–20

- App Development Project 4
- App Design Journal: Prototype

Sessions 21–25

- 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 projects, like exploring augmented reality and virtual reality and hearing from local app designers. 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.

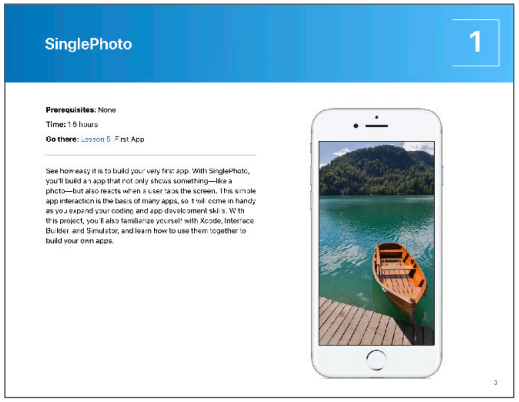
Show off. An app showcase event is a great way to promote your club, app ideas, and coding skills to friends, families, teachers, and the community. It might even help you recruit more members. See page 11 to get tips for holding your own app showcase.

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.

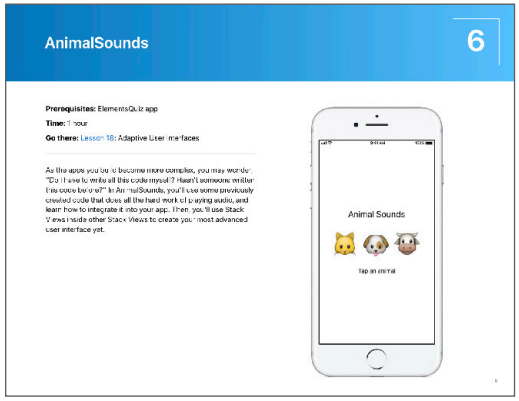
Mix it up. Sometimes members who are more advanced can leave others behind. See if those members can partner up with beginners for pair programming. Teaching someone else is a great way to learn!

Xcode App Development Projects

Eight projects guide a beginner through the basics of coding concepts and app development. Each project builds on the last and results in a working app, focusing on different aspects of apps so that by the end, students have the skills to build their very own apps.



Members aren't expected to complete all eight projects. The goal is to introduce them to different aspects of apps and inspire them to design their own apps.



Members will need the Intro to App Development with Swift course to do the projects. The prerequisites are a must!

Download Intro to App Development with Swift >

Need more information or want to go deeper?

Download Intro to App Development with Swift Teacher Guide >

Tips for Learning with Xcode

Check the console to debug. The console will sometimes have helpful information about what went wrong. The red highlight shows the line in the code where the error occurred.

Stop and think. Bugs are inevitable. Stop and think about the problem. What are its symptoms? Was it working fine until [X]?

There's no one way to write code. Members should review each other's code, provide feedback, and help each other debug.

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

Explore Xcode preferences. Set your text editing and other preferences by choosing Xcode > Preferences from the menu bar. You can add developer accounts, customize navigation or fonts, choose certain behaviors when events occur, and more.

Master keyboard shortcuts.

Build the project: ⌘ B

Build and run the project: ⌘ R

Toggle comments on selected rows of code: ⌘ /

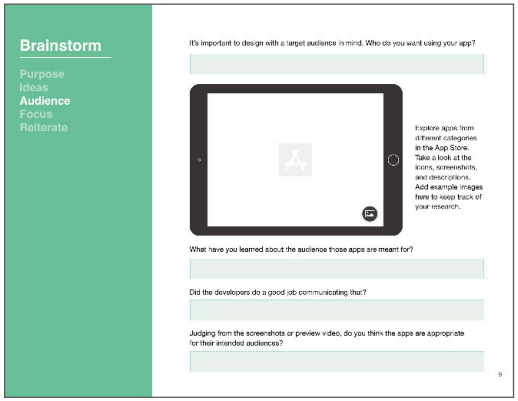
Shift the selected code left: ⌘ [

Shift the selected code right: ⌘]

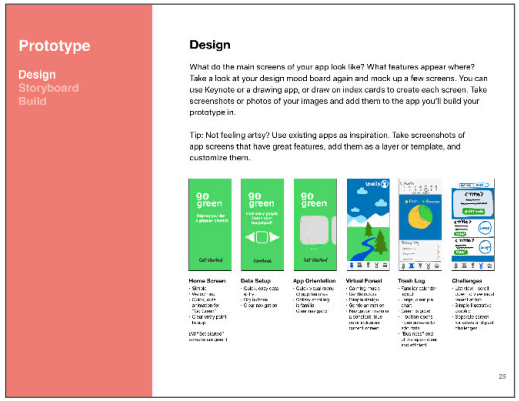
Take it further. Advanced members can work through the App Development with Swift course and focus more on coding aspects of their app prototypes in Xcode.

App Design Journal

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

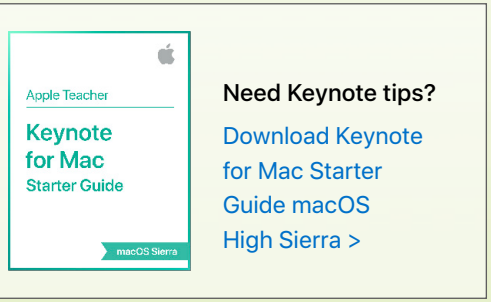


Club members work in small teams to brainstorm and plan the app solution, then create a working prototype of the app in Keynote.



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 Mac Starter Guide macOS High Sierra >

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.

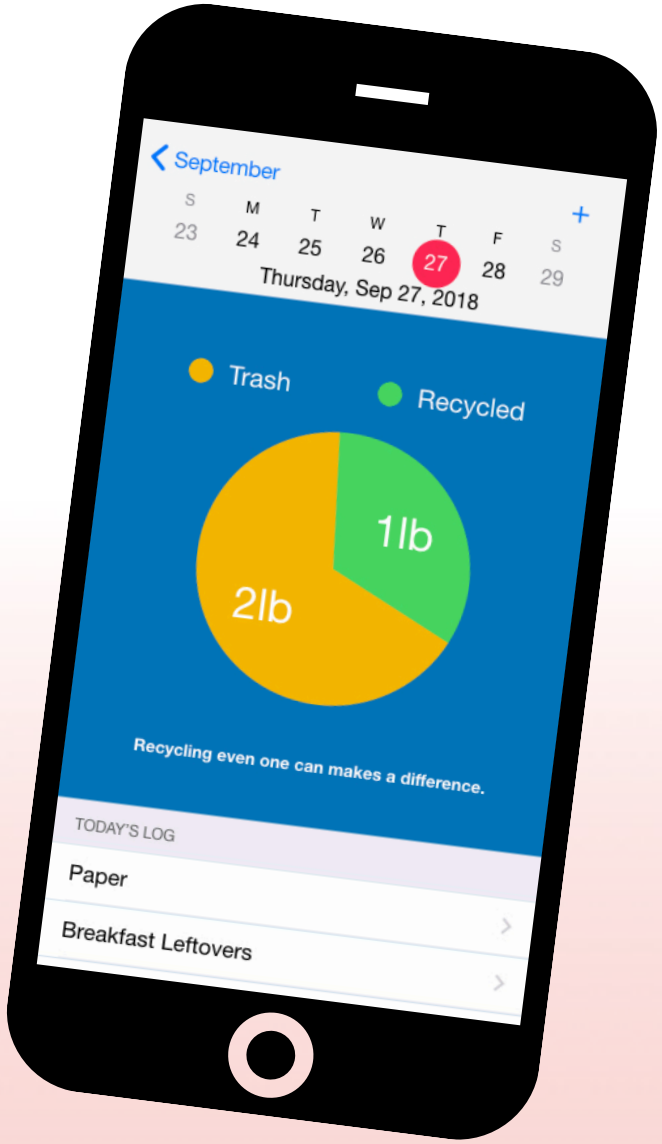
You can download and modify this certificate for different awards.

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.	
Code Implementation (Optional) *Apply this metric if the team has extended their app prototype into Xcode and you're comfortable evaluating Swift and iOS code.*	Code runs with specific examples. It's basic with no abstraction.	Code runs without error in all cases. It's basic and there's some evidence of abstraction.	Code is organized with clear Swift naming conventions and there's high evidence of abstraction. It follows iOS API guidelines and UI conventions.	Code is well documented with comments. There's effective use of applicable Swift language and library features. It employs standard organization principles, such as Model-View-Controller.	
Comments:					**Total score**

Swift Coding Club
Xcode

Certificate of Achievement

Awarded to

For

Signature

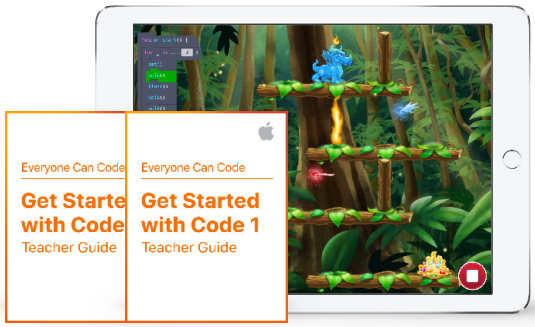
Date

Take It Further

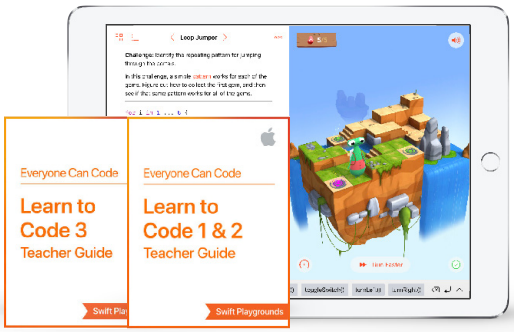
Swift Coding Club is just the beginning of your coding journey. The Everyone Can Code curriculum provides fun, supportive resources to take coders from learning the basics on iPad to building real apps on Mac. App Development with Swift Certification is even available for students who have completed App Development with Swift.

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.

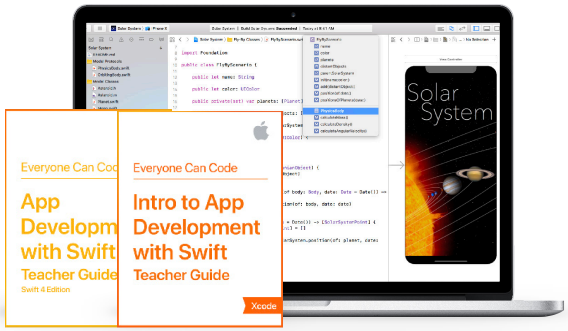
See all the Everyone Can Code resources >



Learn more about the Get Started with Code curriculum >

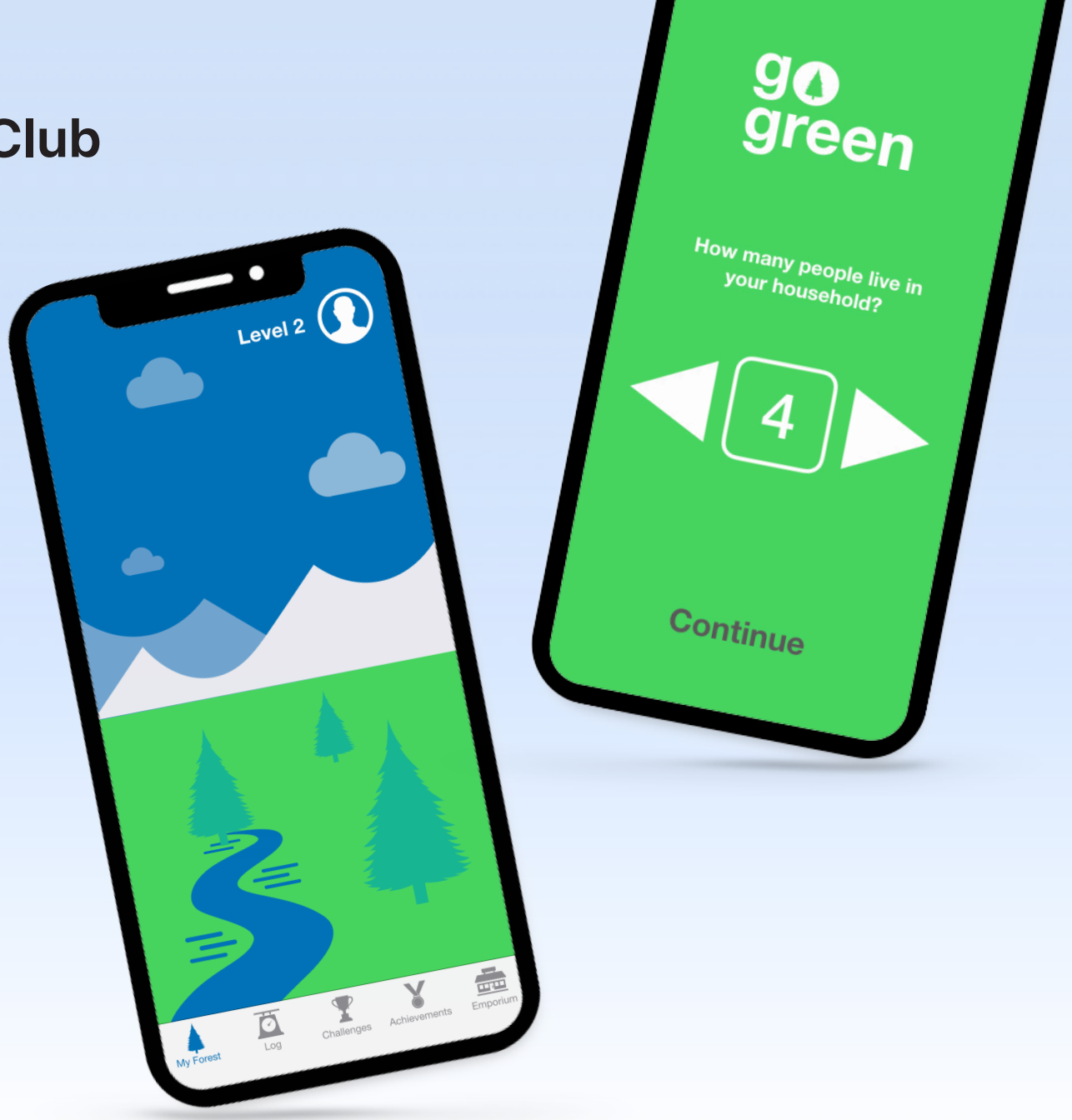


Learn more about the Swift Playgrounds curriculum >



Learn more about the App Development with Swift curriculum >

Swift Coding Club



Xcode

App Development Projects

Projects

Welcome to the Swift Coding Club!

By joining this coding club, you're already on your way to building your first apps. No worries if you're just learning how to code. The projects here will guide you through the basics of both coding concepts and app development. Each project builds on the last, focusing on different aspects of apps, so that by the end you have the skills to build your very own app.

You'll need the Intro to App Development with Swift course to do these projects. Be sure to complete the prerequisites for each one.

SinglePhoto

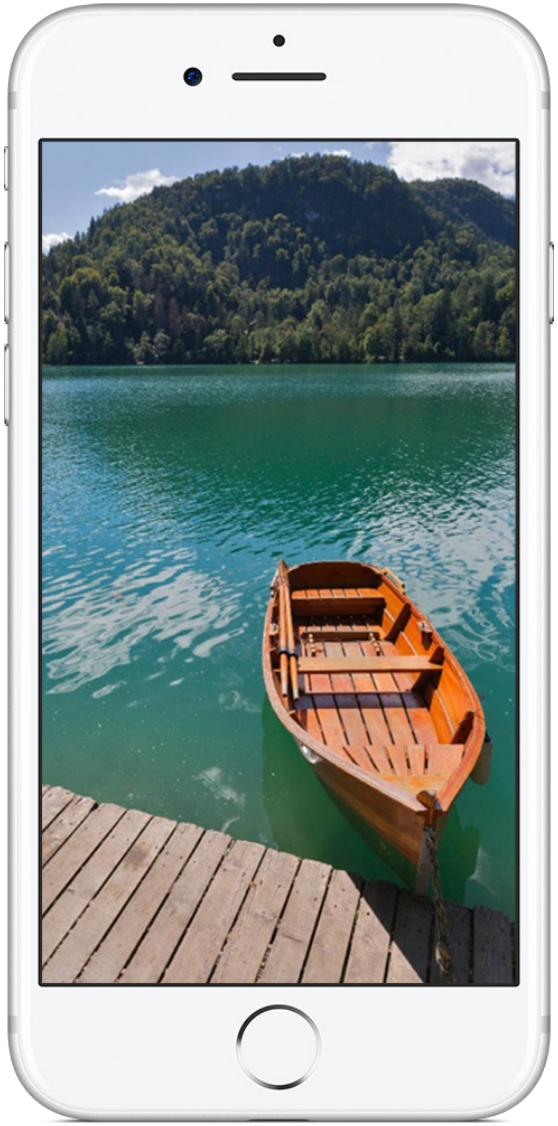
1

Prerequisites: None

Time: 1.5 hours

Go there: Lesson 5: First App

See how easy it is to build your very first app. With SinglePhoto, you'll build an app that not only shows something—like a photo—but also reacts when a user taps the screen. This simple app interaction is the basis of many apps, so it will come in handy as you expand your coding and app development skills. With this project, you'll also familiarize yourself with Xcode, Interface Builder, and Simulator, and learn how to use them together to build your own apps.



QuestionBot

2

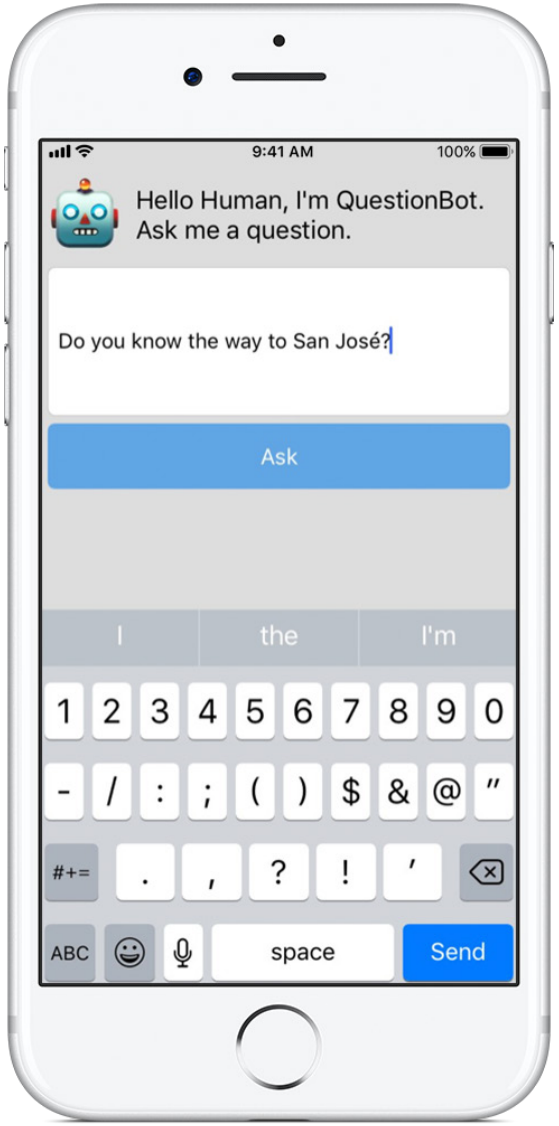
Prerequisites:

- SinglePhoto app
- Lesson 2: Naming and Identifiers
- Lesson 3: Strings
- Lesson 6: Functions
- Lesson 10: Parameters and Results
- Lesson 11: Making Decisions

Time: 6.5 hours

Go there: Lesson 13: QuestionBot

Have you ever used a quiz app or wondered how Siri works? Many apps seem to have a "brain" of their own. With QuestionBot, you'll build an app containing the brains of a bot that responds differently to different questions. To do that, you'll learn how to store values in constants, represent text in code, define inputs and outputs, and more. This project helps you focus on understanding how the app works and the logic that's needed to make your app appear to have a "brain."



ChatBot

3

Prerequisites:

- QuestionBot app
- Lesson 14: Arrays and Loops
- Lesson 15: Defining Structures

Time: 3.5 hours

Go there: Lesson 16: QuestionBot 2

Now let's make QuestionBot even smarter. You want to use your app over and over again, and it would be helpful if it could "remember" your past interactions. With ChatBot, you'll build an upgraded version of QuestionBot that retains a history of the messages between the user and the bot. ChatBot is already partially built, so you can concentrate on learning the various skills needed to build the part of the app that keeps track of the conversation.

ColorMix

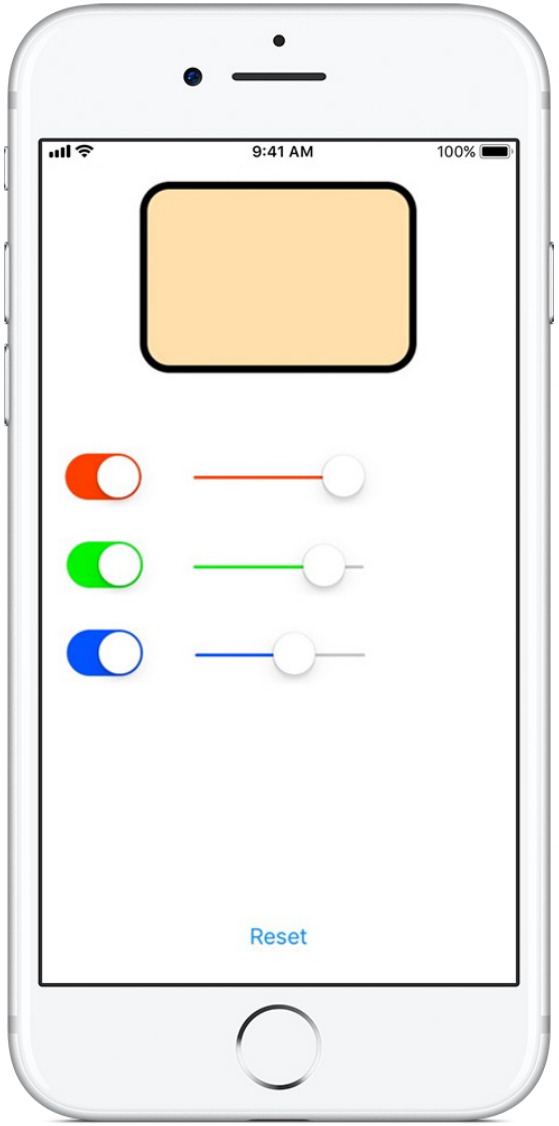
4

Prerequisites: ChatBot app

Time: 1.5 hours

Go there: Lesson 17: Actions and Outlets

Think about the user interface (UI). So far, you've built apps where a user can tap buttons and enter text. With ColorMix, you'll also learn how to add switches and sliders to the UI. More important, you'll learn how to connect those visual UI elements to Swift code so they work the way you want them to. In the end, you'll have ColorMix, an app that generates colors by mixing red, green, and blue.



ElementsQuiz

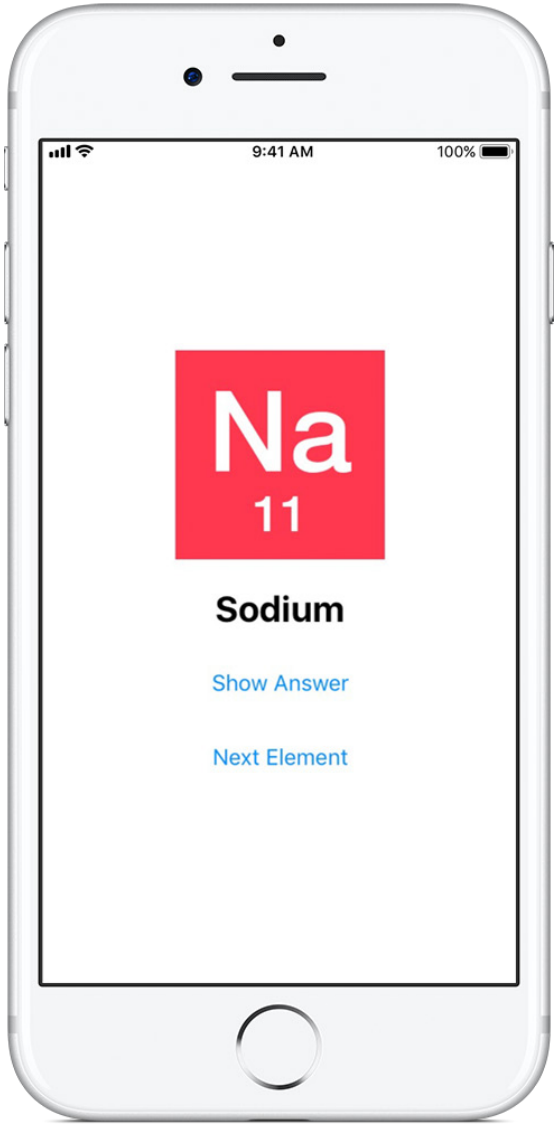
5

Prerequisites: ColorMix app

Time: 2 hours

Go there: Lesson 18: Adaptive User Interfaces

Most people download apps to solve a particular problem—to help them get organized, calculate their finances, or get directions, for example. In ElementsQuiz, you'll build an app that helps students memorize elements of the periodic table. You'll use Stack Views to easily position multiple views on the screen at once, and the positions will adjust to fit a wide variety of screen dimensions.



AnimalSounds

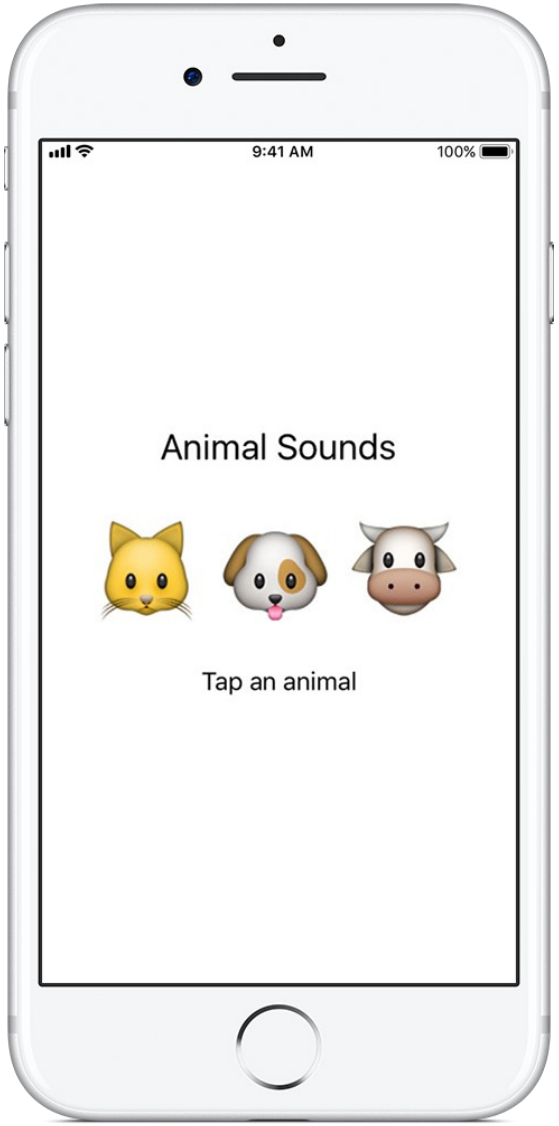
6

Prerequisites: ElementsQuiz app

Time: 1 hour

Go there: Lesson 18: Adaptive User Interfaces

As the apps you build become more complex, you may wonder, "Do I have to write all this code myself? Hasn't someone written this code before?" In AnimalSounds, you'll use some previously created code that does all the hard work of playing audio, and learn how to integrate it into your app. Then, you'll use Stack Views inside other Stack Views to create your most advanced user interface yet.



RockPaperScissors

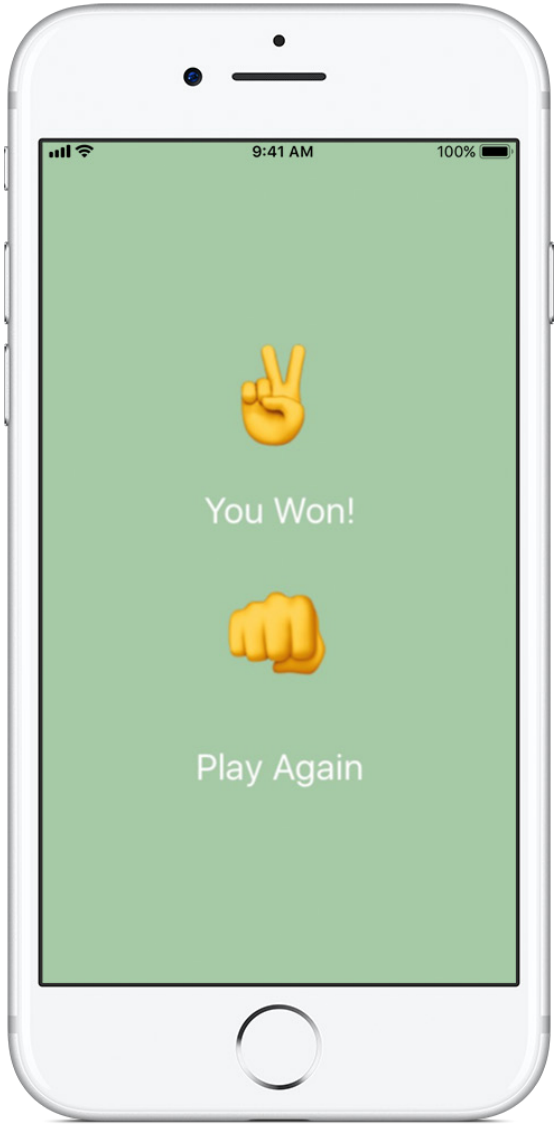
7

Prerequisites:
- AnimalSounds app
- Lesson 19: Enumerations

Time: 1.5 hours

Go there: Lesson 20: Final Project

You'll often need to present the user with a list of choices, such as a direction to travel or a type of cuisine. In this project, you'll learn how to create the three unique options for a game of Rock, Paper, Scissors so that the user can play against the computer endlessly. Along the way, you'll learn how to generate random numbers so that the computer selects a random choice.



MemeMaker

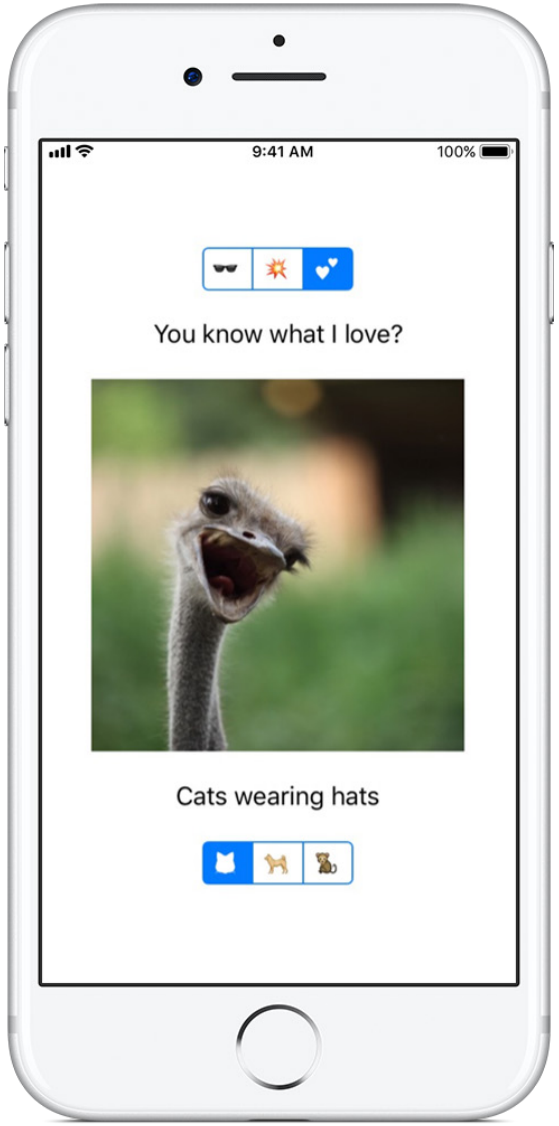
8

Prerequisites:
- AnimalSounds app
- Lesson 19: Enumerations

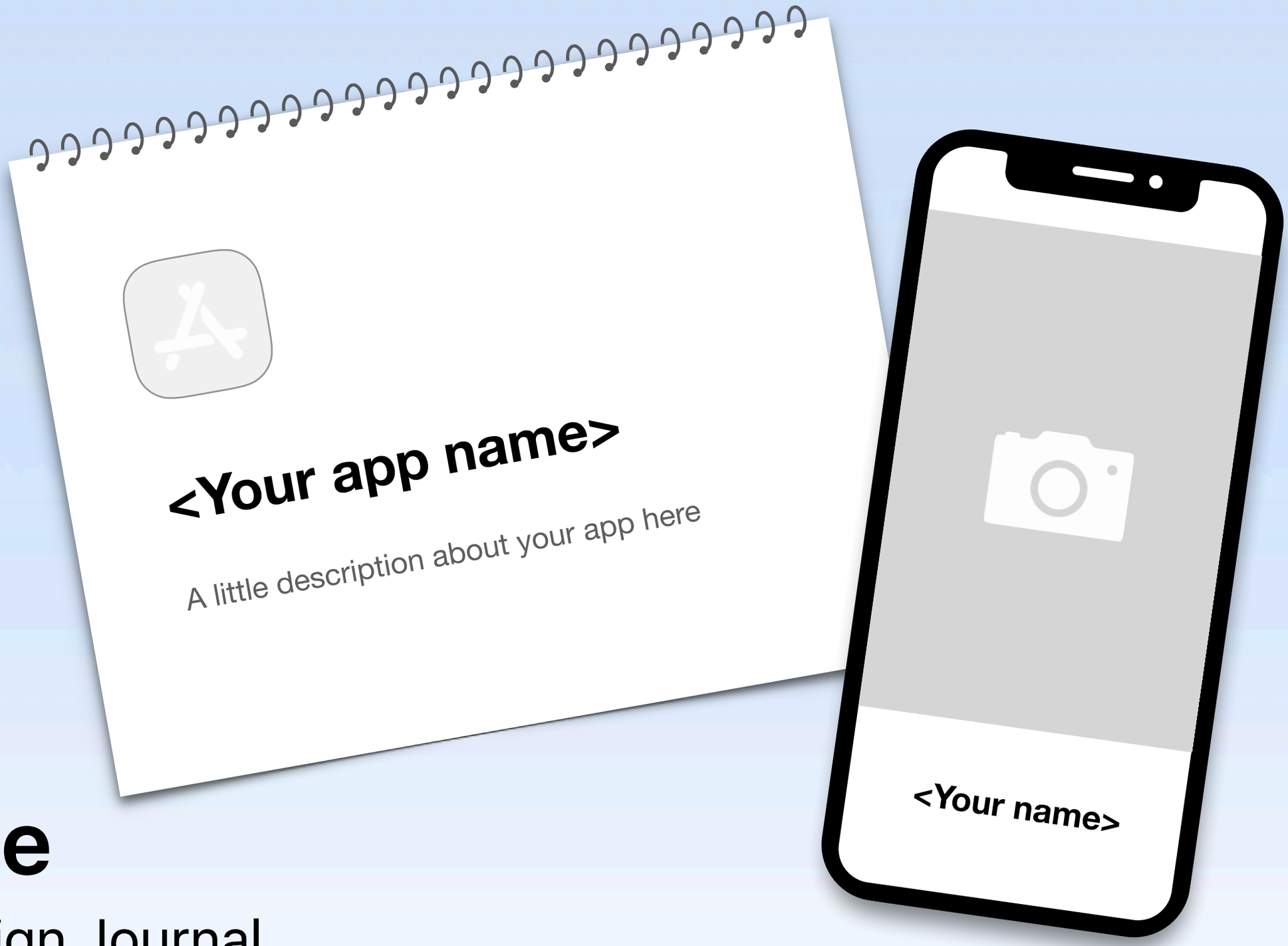
Time: 1.5 hours

Go there: Lesson 20: Final Project

An Internet meme is typically a still image with text at the top and bottom. In MemeMaker, you'll learn to use Segmented Controls to display different captions above and below an image. Since the controls are independent, you can mix and match the text to create custom combinations.



Swift Coding Club



Xcode

App Design Journal

Welcome

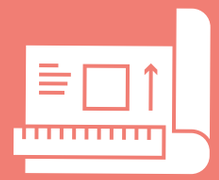
Your Keynote app design journal will help you keep track of your ideas and guide you as you cycle through the four stages of the app design process. You can play it through to see what's in it, but you'll work in slide view to add notes, images, shapes, and more. Feel free to add and duplicate slides,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

My app idea

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

App name:

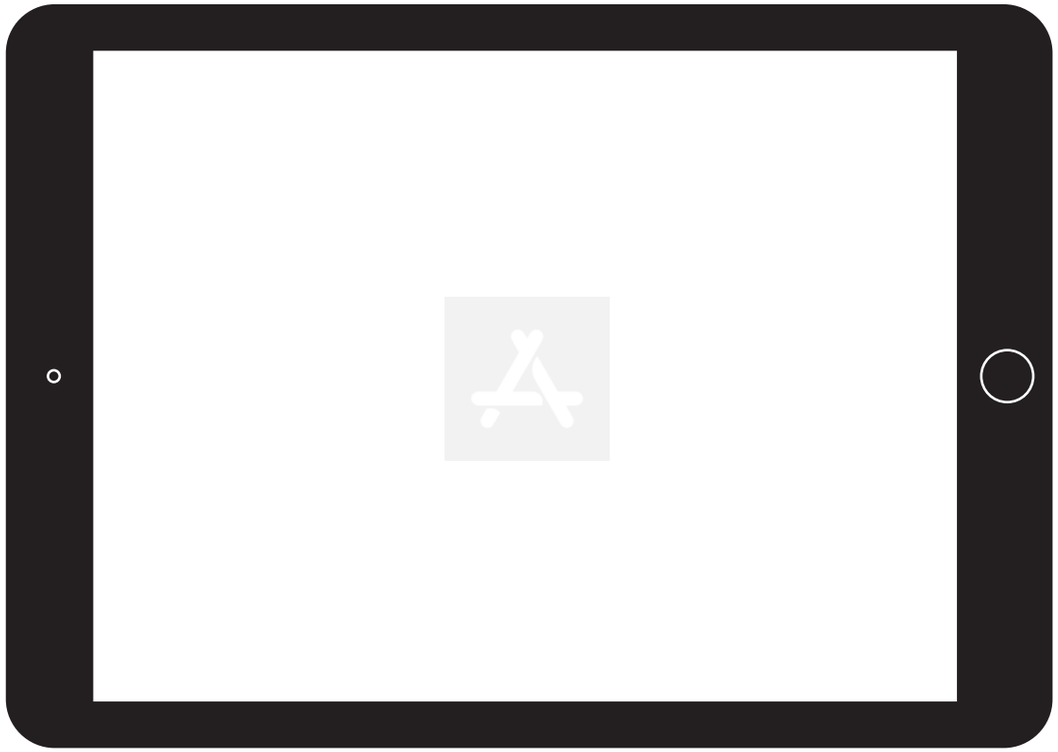
Write a description of what the app does.

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?

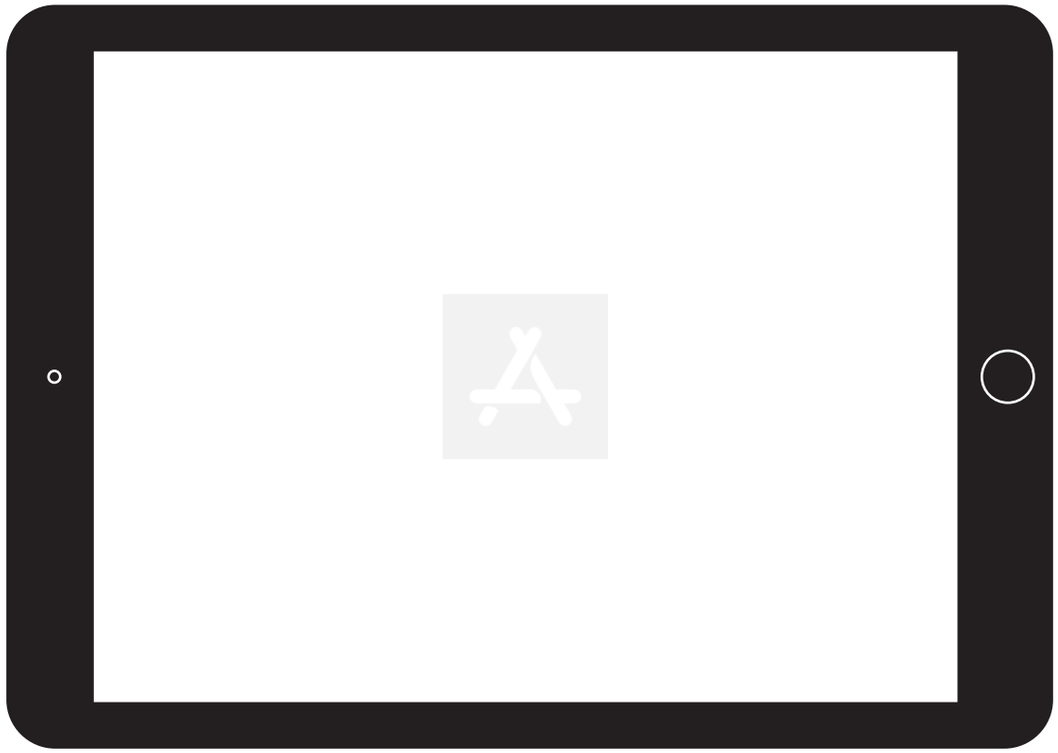
Can you suggest improvements to the user interface?

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?

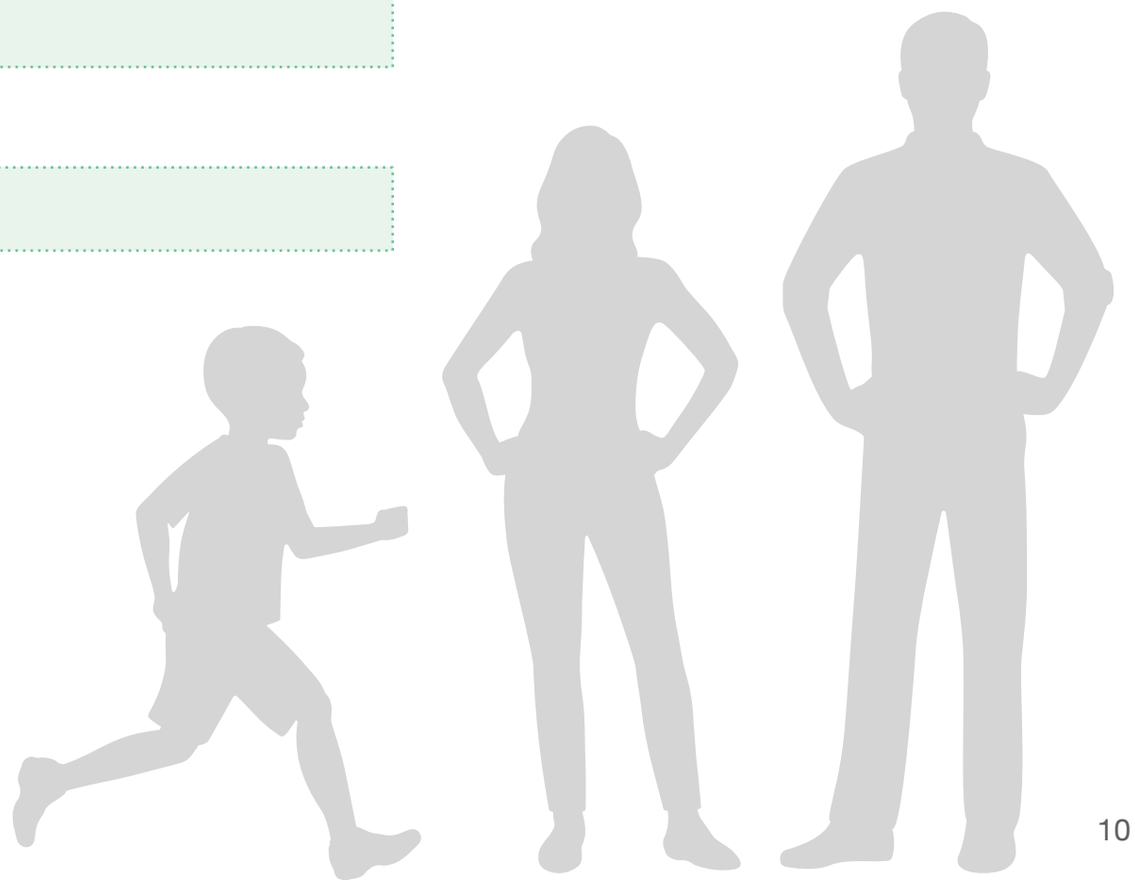
Why is the person using the app?

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

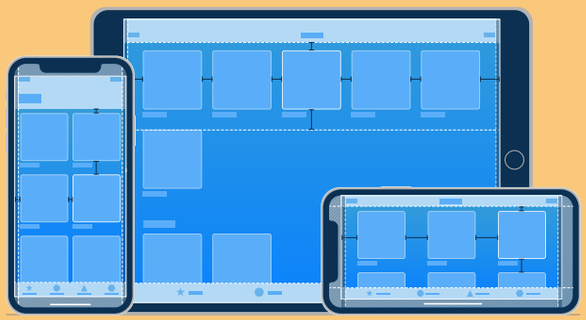
Overview

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

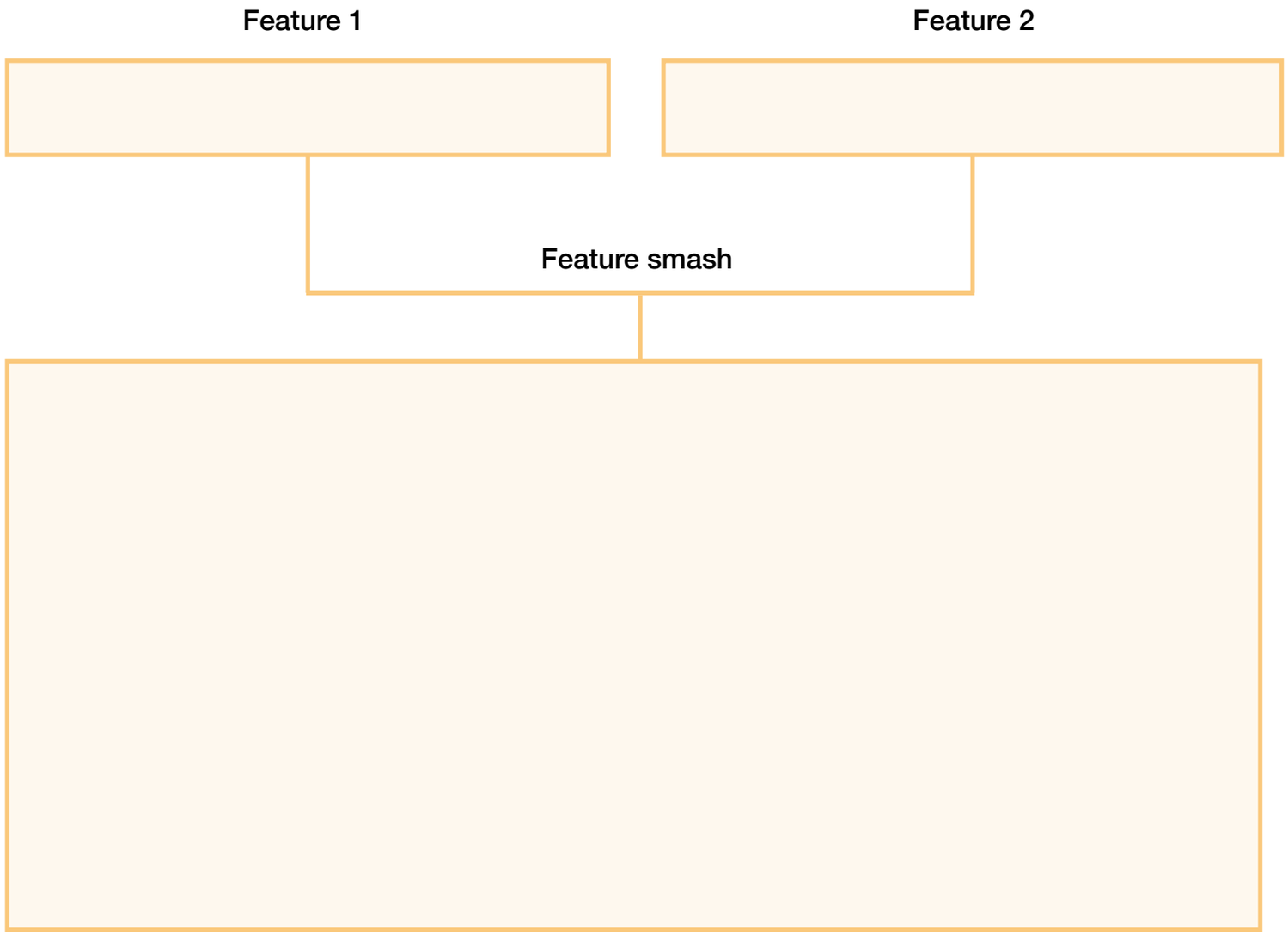
iOS supports multiple ways to access onscreen content. People with physical impairments can use Siri or Dictation to interact with apps. People with visual impairments can increase the size and contrast of screen elements, or use the VoiceOver screen reader and navigate your app entirely by sound. But 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

Deeper Dive: The V in MVC
Model View Controllers (MVCs) are design patterns that organize an app's files, structures, and classes. Learn more in Unit 4, lesson 3.

Now that you've decided on the features, you need to think through exactly how, when, and where your app will use these features and how you can use code to achieve it. It's important not to get bogged down in the details of data modeling or how many controllers you'll need for a particular feature. For now, keep it simple: Make some simple sketches on paper, focusing on key views that are important to your app.

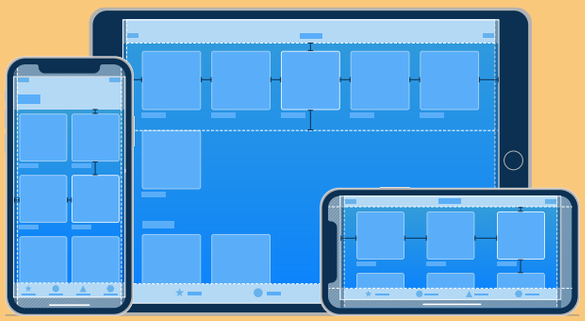
What views do you need to create a feature in your app? Draw or mock up the key views. As you mock up the views, think about:

- How will you display the data of your app?
- How are you capturing input from the user?
- Are there particular views you want to show on multiple screens?

Use your answers to these questions to explore alternative input methods, to determine if you're using too many different controls, and to locate views that you'll want to make reusable in code.

Plan

UI/UX
iOS features
Design



Be sure to review the Human Interface Guidelines again.

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?

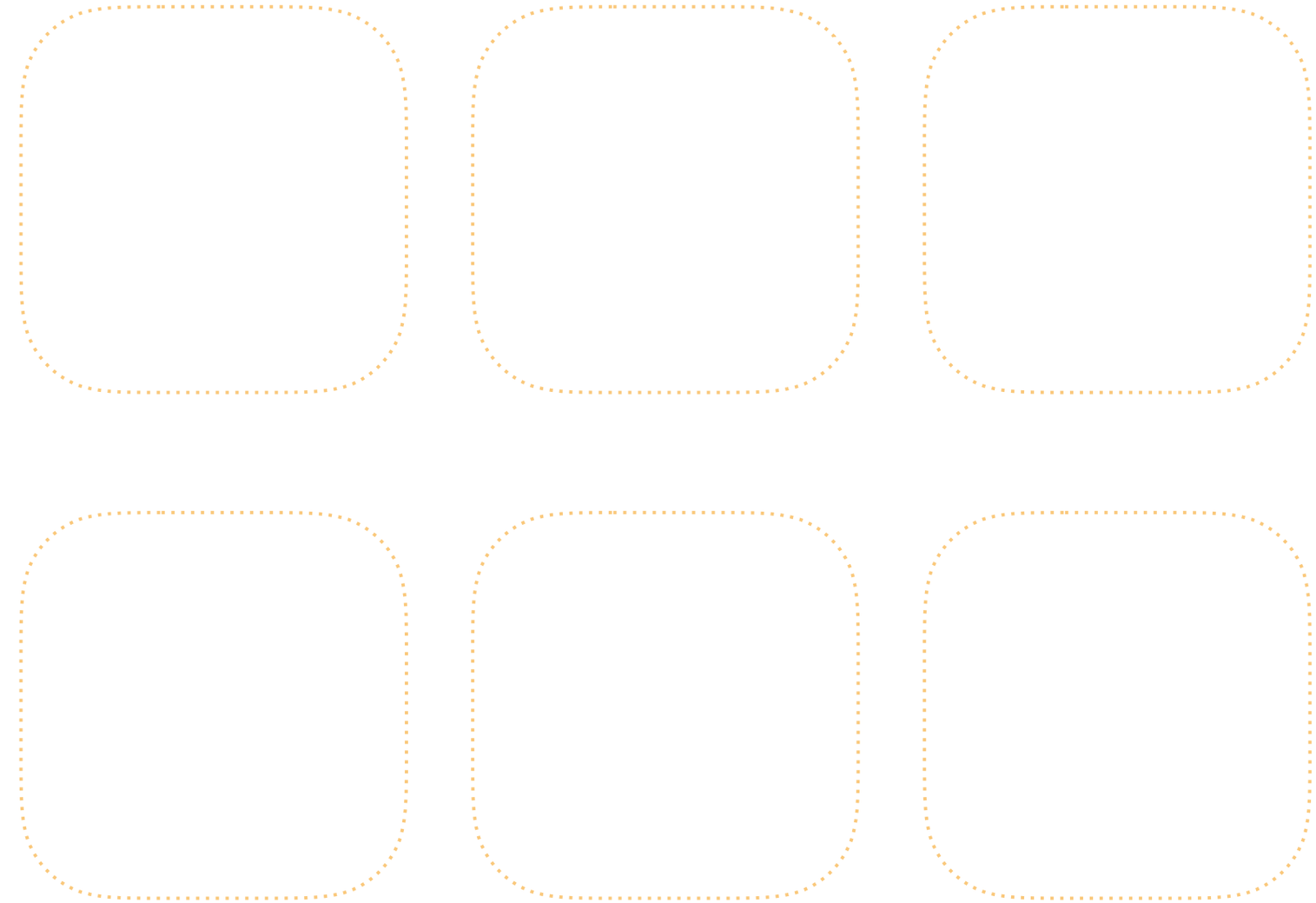
Add phrases of design principles, such as "Keep it simple," "Modern," and so on.

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.

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.

Home Screen

- Simple
- Welcoming
- Quick, cute animation for "Go Green"
- Clear entry point to app

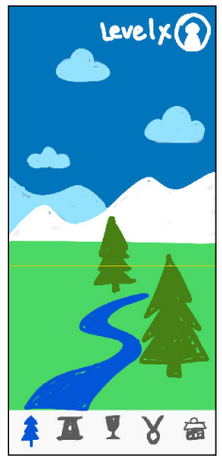
(All "Get started" screens are green)

Data Setup

- Quick, easy data entry
- Big buttons
- Clear navigation

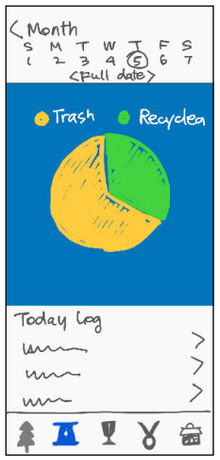
App Orientation

- Quick visual menu of app features
- Gallery scrolling is familiar
- Clear navigation



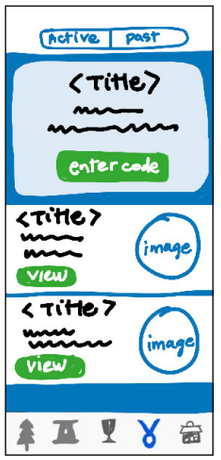
Virtual Forest

- Calming music
- Gentle colors
- Simple design
- Gentle animation
- Navigation menu is a constant; blue color indicates current screen



Trash Log

- Familiar calendar layout
- Large, clear pie chart
- Green is good
- + button opens more screens to add data
- "Business" end of the app—clear and efficient



Challenges

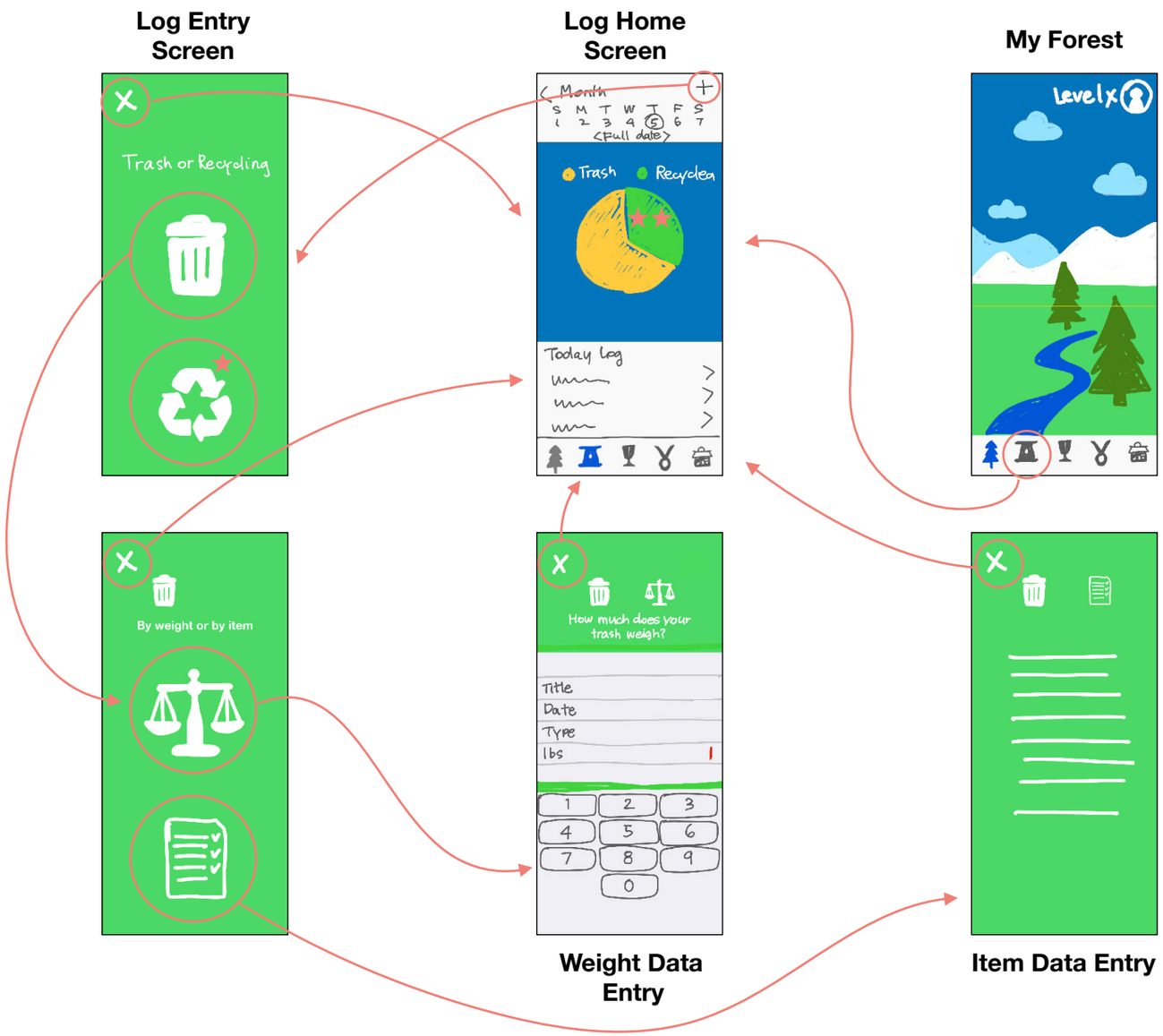
- List view—scroll down to view most recent at top
- Simple illustrative graphic
- Separate screen for active and past challenges

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. Click the Document tab in the Document sidebar, click the Slide Size pop-up menu, 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. Control-click the object you want to link, choose Add Link, then choose Slide.
5. To make sure that the presentation changes slides only when the user taps the navigation buttons, click Document, then under Presentation Type, choose Links Only.

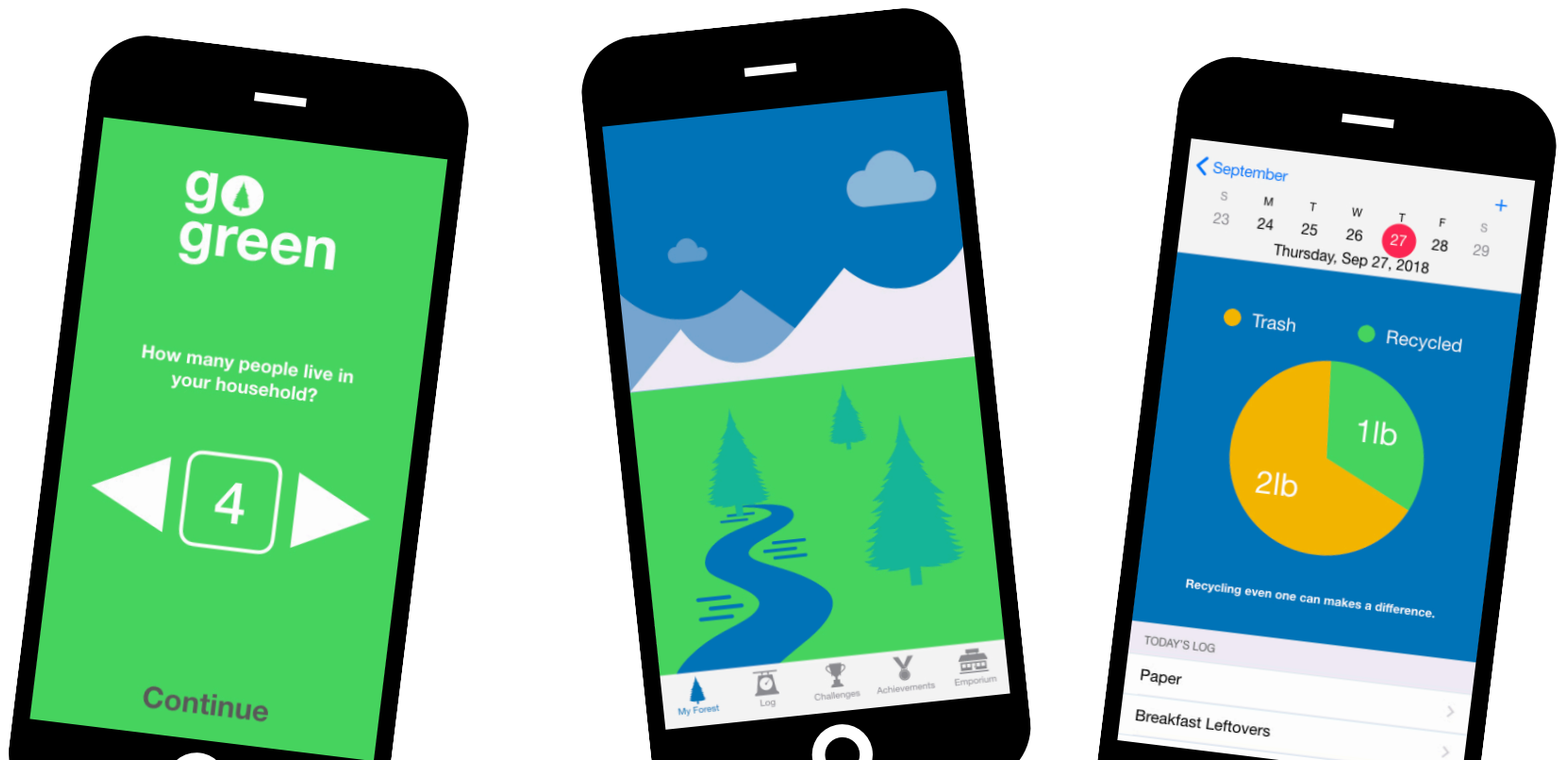
Build

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.

As you experiment with your own UI, you might also want to design apps that match the iOS design language. Download this Keynote document to access iOS UI elements.

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?



Deeper Dive: Detailing MVC
Once you've defined how your app will flow in your prototype, think about how to organize different parts of your code. The Model-View-Controller pattern helps accomplish this while keeping your code organized.

Use the following questions to help build a list of models, views, and controllers that your app would need if you began building the prototype with code.

Models
What data do you need to build your features?
Where does the data come from? Does the user supply it or does it come from a web service?
Do you need to store the data on the device for offline access?

Views
Are there particular views you want to show on multiple screens?
Did you include any custom gestures?

Controllers
How many view controllers does your app need?
What controllers will help manage the data?
Does your app have custom transitions that need a controller?

Deeper Dive: Using Xcode
One of the benefits of using simple prototyping tools, such as drawing and Keynote, is that it's very easy to make changes based on feedback. However, there's no rule that says you can't build portions of your prototype in an Xcode project, using Interface Builder and storyboards to piece together all the screens in your app. When should you consider reaching for an advanced developer tool for prototyping?

Keynote prototyping is great for communicating the UI of your app, but it's difficult to emulate all UX elements, especially app interactivity and feedback. Building parts of your app in Xcode enables you to demonstrate how the app responds to genuine user interactions. In your Xcode project, you might choose to focus on coding just one particular app function—you don't need to build the whole app or have perfected the UI.

What sections of your app might you want to write code for first? Outline a plan of attack to build your app. Look back at your storyboard.

Which views are most important that they work first?

Which views might you want to save to build at the end?

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?

Brainstorm

Purpose
Ideas
Audience
Focus
Reiterate

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