

HOPSCOTCH



Teacher-led Lesson Plan

TIME

45-60 minutes (+15 minutes of optional, free code time)

BIG IDEA

Coding is a superpower! If you know how to use individual blocks like conditionals and variables, you can put them together in powerful ways to make what you want. Little blocks build big programs!

SKILL FOCUS

-
- Planning and Execution
 - Planning and carrying out investigations (NGSS Practice 3)
 - Analyzing and interpreting data (NGSS Practice 4)
 - Reason abstractly and quantitatively (CCSS.MATH.PRACTICE.MP2)
 - Look for and express regularity in repeated reasoning (CCSS.MATH.PRACTICE.MP8)

KEY VOCABULARY

Sequence: The order in which instructions are given to the computer

Event: When something happens

Rule: Instructions that tell your computer what to do (the command) and when to do it (the event)

Value/Variable: A number that can change

Conditional: statement of the form "IF (something is true) THEN (do an action)"

TRANSFER GOALS

-
1. Students will understand that a conditional checks if something is true and can identify the use of conditionals in their lives.
 2. Students will understand that a value or variable is a holder for a number and can identify the use of values in their lives.
 3. Students will become comfortable with keeping track of variables and doing operations.
 4. Students will make a plan and see it through until the end.
 5. Students will recognize patterns in their code and apply them to new situations.

MATERIALS

– 1 iPad or iPhone per student, or 1 device per 2 students, for pair programming

– Complete project available:

<http://hop.sc/1L5sH2Q>



TEACHER BRIEF

Hi!

We're *really* excited that, this Hour of Code, you're programming with your students—both for them and for you. Kids have remarkable imaginations, and creating computer programs is an amazing way for them to express themselves. We've seen kids create astonishing things using our simple but powerful tool.

Anyone, regardless of their experience with programming, can teach this lesson. Just as Hopscotch was built on the principle that anyone can become a great programmer, this lesson is designed on the premise that anyone can become a great programming teacher.

In this lesson, students will create a if-you-chose-mostly-As type quiz. In our example, it will help them answer an age-old question: "Which emoji are you?" **With a little adjustment, it could be made into a factual quiz that checks the answers for correctness; for example, you could adapt it for an english class to test mastery of literary techniques or in biology to assess knowledge of cell functionals.** This lesson introduces some of the most important computing concepts: sequencing, events, loops, values, and conditional statements. You can find a glossary of terms at the end of the lesson (page 13).

This HOC activity is broken into smaller tasks, each of which begins with a discussion of the programming task at hand and the core concepts evoked in completing it. You can use this context to lead conversations with your students or, if they have prior programming experience and a good grasp of these concepts, feel free to skip.

After a discussion of what needs to be built and, if desired, how it might be coded, students can start coding. Depending on how many devices you have, you can have students work independently or in pairs. At Hopscotch, we do a lot of pair programming (where two engineers share one computer) because it helps us write smarter, less-buggy code. We recommend trying it! All students should get into the habit of testing their code frequently by running (playing) it. It is much easier to find and solve mistakes when you're constantly testing.

We demonstrate how each task can be implemented in Hopscotch with screenshots of sample code. The code we suggest usually is only one way to build the needed feature; there are often other ways that students can accomplish their goals.

Have fun and we can't wait to see what your students build. Share their projects on social media and tag with #madeonhopscotch and @hopscotch on Twitter and @gethopscotch on Instagram :-)

Yours,
Jocelyn Leavitt
Co-founder and CEO, Hopscotch



LESSON

0. Make a plan

The first step is to design the questions and answers for your quiz. Pass out copies of the blank matrix below, or put it on the projector and have students create their own on paper.

	Question 1	Question 2	Question 3
Result A	Answer 1A	Answer 2A	Answer 3A
Result B	Answer 1B	Answer 2B	Answer 3B
Result C	Answer 1C	Answer 2C	Answer 3C

You can have students create a quiz tied to a book they've read, their Spanish vocab words, or what they're learning in Science class. If time is limited, or if a student has a hard time coming up with an idea, use the sample matrix below:

	What are you best at?	Where do you like to hang out?	Would you move to Mars if you could?
			
	Playing music	Library	No.
	Playing sports	Park	Yes!
	telling jokes	Pool	Depends, do they have ice cream?

1. Setup and introduction to variables.

Before coding, discuss how the quiz will work and the tools we will use to build it. The quiz has three main functions: First, it keeps track of what question you're on and advances the question number when the player chooses an answer. Second, it keeps track of how many of each answer you've chosen. Third, it compares the answers at the end to give the player a result. We will use **variables** to keep track of this information.

Variables hold pieces of information. Variables can be set, changed, or checked. They can be used inside **events** and other blocks, and can stand in any place you can use a number. What makes variables so powerful is that you can actually change them programmatically. For example, think of your score in a game like Angry Birds or the number of unread emails in your inbox. Both of these numbers are actually variables. As you score more points in Angry Birds your score goes up; the variable changes. As you read your email the number of unread emails you have goes down. Variables are used to represent some kind of information.

In Hopscotch, the yellow Variables tab is to the right of the calculator tab that pops up when it's time to input a number. In addition to the built-in variables like "iPad's width" or "Character's X position", you can add new variables to your project to keep track of other things like a score or a state. It might be helpful to walk your students through the



LESSON

It is super important to name your values well! You'll need to be able to tell them apart later and remember what they're for! If you need to rename or delete a value, press and hold down on it.

After discussing the concept of values with your students, you might want to go through a list of objects that you will need and their respective initial rules (five text objects: a title, a question, and three answers.) The title (or any object, really) should initialize a value that will keep track of what question the player is on. In the example, we call it "Question Number". We create separate values that will keep track of each answer category. We initialize "Question Number" to 1 and the three "Answer" values to 0.

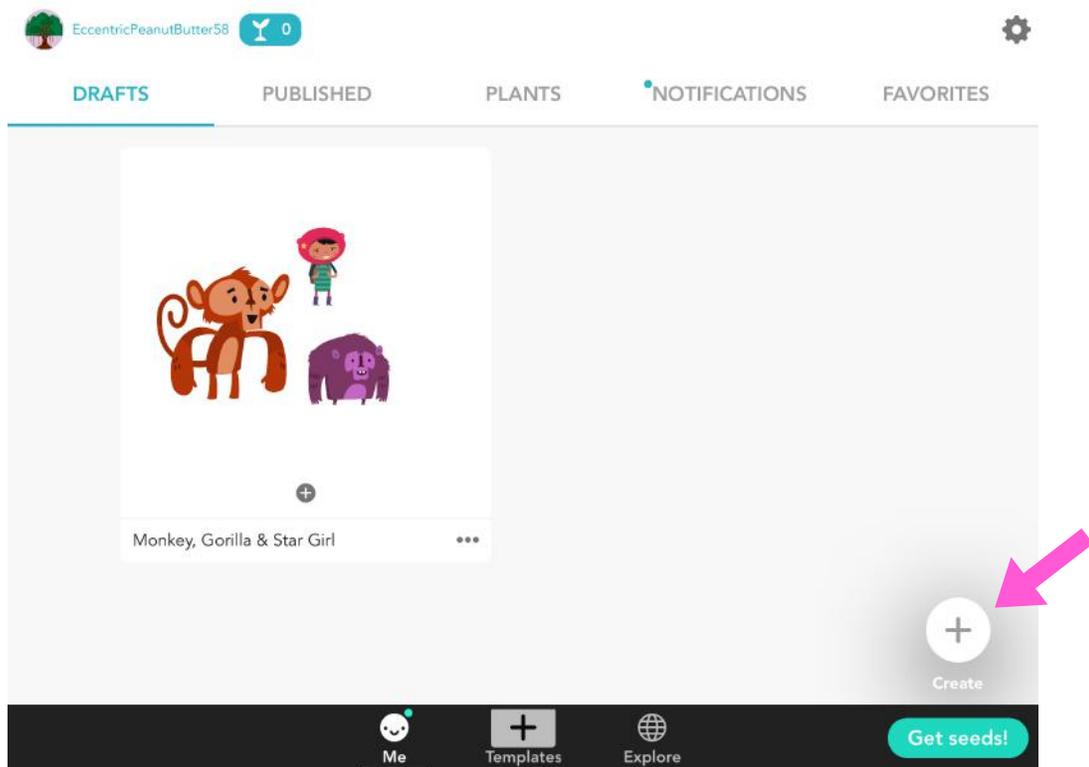
1. Using Hopscotch (5 minutes)

Download and get your students acquainted with Hopscotch. (<http://hop.sc/gethopscotch>)

1.1 Finding the Hopscotch app on your iPad or iPhone

1.2 Signing into your account (please write to us to request student accounts at hello@gethopscotch.com.)

1.3 Making a new project: Tap on the white circle "+" button labelled "create" screen.



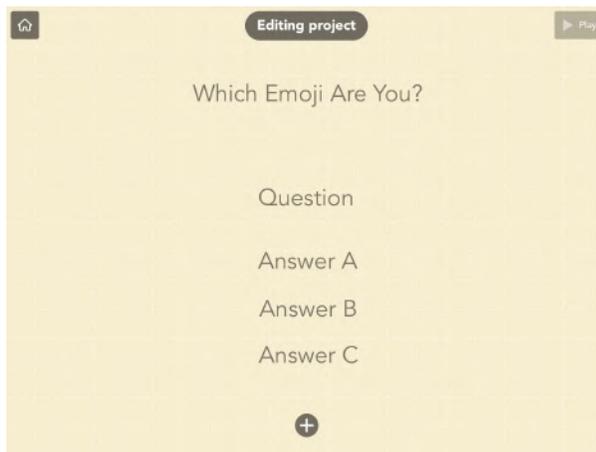
LESSON

2. Set up (5 minutes)

The following instructions explain how to build a sample quiz that answers the question “Which emoji are you?” You can swap in your own questions and answers in your quiz. The player will navigate through the quiz by tapping on questions and answers that appear as the player advances.

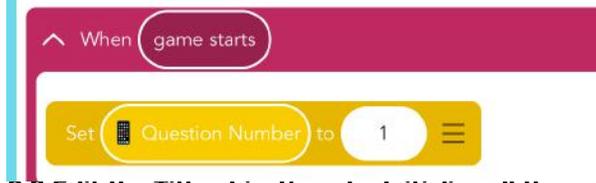
One of the most important lessons of this activity is learning that the programmer must not only put together all the components of a game (buttons, background, character), but also

2.1 Add text objects for the title, question, and three answers (5 objects)



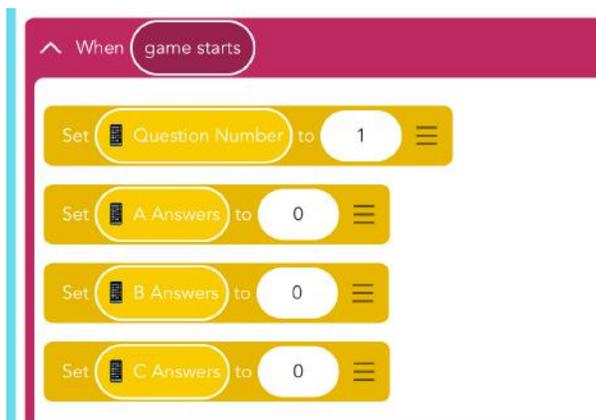
Name each text object according to the role it will serve in the project, e.g. “Title” or “Answer A/B/C”.

2.2 Add new code to the Title object: Initialize Question Number to 1



Select “New Variable” under the Game icon and name your new variable “Question Number”.

2.3 Edit the Title object's code: Initialize all three answer variables to 0.



Create a new variable for each Answer (e.g. A Answers, B Answers, C Answers.)



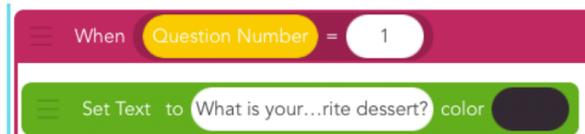
LESSON

3. Value Events (EV)

The four events at the very bottom of the Events menu deal with comparing values. They track whether a value is equal, not equal, bigger than, or smaller than another value or number. When you select the “_ equals _” event, the calculator menu opens. On the right of “Calculator” are values.

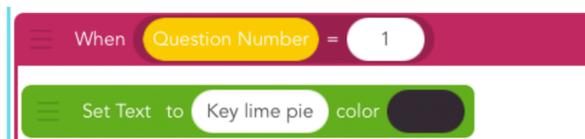
The quiz should display the first question and its answers when the “Question Number” value equals 1. The player chooses an answer and the game should keep track of how many of each kind of answer is chosen. We do that by increasing the appropriate answer value when each answer is tapped. For example, if an “A Answer” is chosen, the value “Answer A” should increase. Show the students these new events, and then discuss the rules that they will need to code before having them start programming. Make sure they choose the right object-value combination!

3.1 Add new code to the question object



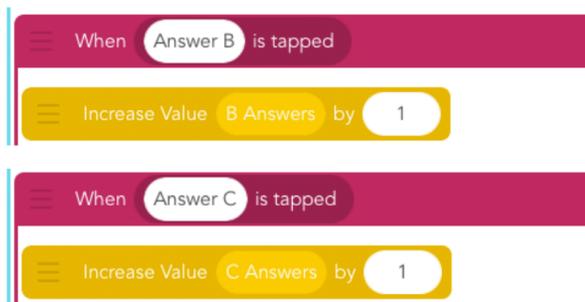
Type the first question in the “Set Text” window.

3.2 Add new code to each of the answer objects



Refer to your question matrix to enter each answer in its respective rule. We will do questions 2 and 3 later.

3.3 Add new code to each of the answer objects: Increment the right score when tapped



Tap the iPhone icon in the rule's header to select the corresponding text object.



LESSON

4. Show Values (LV)

It's hard to know whether our values are working properly, so programmers design tests to display them. In Hopscotch, you can do this by displaying the value on the screen. This is an advanced debugging strategy, and the values should be deleted from the finished version of the project once you know everything works. Discuss this technique with students and see if they can figure out how to display a value when their project is played.

We recommend using "Set Text" and putting the appropriate value inside a forever loop to display it when the project is played. You should see the values change as the player progresses through the quiz.

4.1 Add four new blank text objects. Place them next to the question and each of the answers.

4.2 Add new code to the question test object



To set text to the value Question Number, tap the three gray bars above "Q" on the keyboard.

4.3 Add new code to Answer A test object



4.4 Add new code to Answer B test object



LESSON

4.5 Add new code to Answer C object

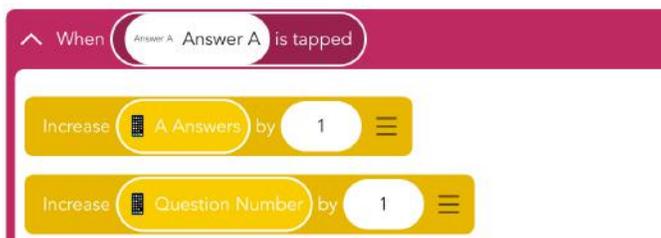


5. Next Question (EV)

The score for each of the answers advances as you tap them, but the question does not change (neither does the "Question Number" value). Decide as a class, what event(s) should trigger the quiz to move on to the next questions. If we make "Question Number" increase, we also need to make the question (and answers) display the correct text for when the "Question Number" value equals 2 or 3. How should this work?

Hint: You advance "Question Number" when any of the answers is tapped! We have already made rules for this event, so we just need to add a block to increase the "Question Number" value. You can do this step as a class and then together or in pairs test your programs to make sure the "Question Number" is advancing. Allow your students to implement the answer code in their own time and to help their neighbors if they finish early.

5.1 Edit code for all 3 answer objects to change "Question Number"

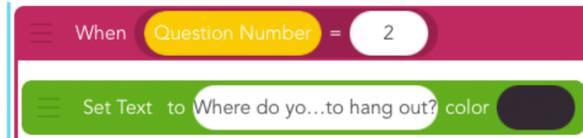


Add the Increase Variable block to Answers B and C.

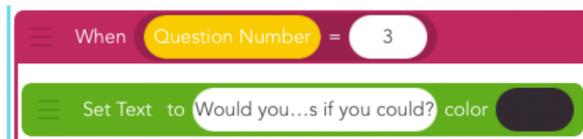


LESSON

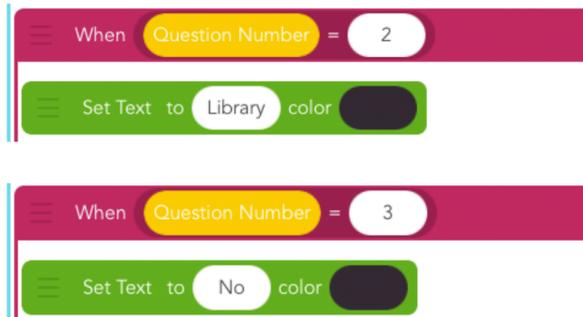
5.2 Add new code to Question object to display second question



5.3 Add new code to Question object to display third question



5.4 Add new code to each Answer object to display second and third answers



For Answer objects A, B, and C, add these same two rules with their respective answers. Refer to your matrix for the text of the answers. In this step, you're creating a total of 6 new rules!

6. Results and introduction to Conditionals (VC)

Now that all the questions are answered, we have to check our results! The correct result to show is the one for which the "Answers value" is the greatest. We can use **conditionals** to compare values.

Conditionals are statements of the form "IF (something is true) THEN (do an action)". It executes code only under the *condition* that you specify, like IF the score is greater than 10 or IF a character is invisible. We use conditionals in our lives all the time. Stoplights are a great example of conditions: "If the light is green, then go." They are also extremely important in programming since computers need explicit directions.

There are two types of conditionals in Hopscotch: "Check Once If" and "Check If... Else".



LESSON

“Check Once If” is used when there are only two possibilities: 1. If the condition is true, do something, and 2. If the condition is not true, move on. “Check Once...Else”, by contrast, lets you check two things before moving on: 1. If something is true, do something, and 2. If this thing is NOT true, do something else, then move on. For example, say you can go in the ocean only if you’re wearing sunscreen or else you will have to put some on first. If you are wearing sunscreen (the condition is true), then you may go into the ocean. If you are not wearing sunscreen (the condition is false), then you must put some on first.

There are several ways to determine the answer to the quiz. The safest way is the one that won’t have to be changed if more questions are added: A is the winner if the “A answers” value is greater than *both* the “B answers” and the “C answers” values (and likewise for B and C.) In Hopscotch, we check if two things are true at the same time by using the “and” expression under Conditionals after choosing Check Once If.

6.1 Add new code to the Title object

The image shows a vertical stack of code blocks in a purple-themed environment. The top block is a 'When' block with a yellow 'Question Number' block and a white '3' block. Below it are three 'Check Once If' blocks, each with two yellow comparison blocks and an 'and' connector. The first 'Check Once If' block has 'A Answers > B Answers' and 'A Answers > C Answers'. Below it is a green 'Set Text' block with 'You are the 🕶️ color' and a dark grey color picker. The second 'Check Once If' block has 'B Answers > A Answers' and 'B Answers > C Answers'. Below it is a green 'Set Text' block with 'You are the 😍 color' and a dark grey color picker. The third 'Check Once If' block has 'C Answers > A Answers' and 'C Answers > B Answers'. Below it is a green 'Set Text' block with 'You are the 😬 color' and a dark grey color picker.

LESSON

6.2 Edit the Title object's code in case there's a tie (optional)

```
Check Once If C Answers > A Answers and C Answers > B Answers
Set Text to You are the 😏 color
Check Once If A Answers = B Answers and A Answers = C Answers
Set Text to It's a tie!!! color
```

7. Tidying up (ES)

The hard part is done, and there's just a bit more work to do to make the ending look snazzy. One option is to center the results and hide the question and answers at the end. Your students may come up with better ideas!

7.1 Edit Title's code

```
When Question Number > 3
Set Position to x 500 y 400
Check Once If A Answers > B Answers and A Answers > C Answers
Set Text to You are the 😏 color
Check Once If B Answers > A Answers and B Answers > C Answers
Set Text to You are the 😏 color
```

7.2 Add new code to the question and all three answers

```
When Question Number > 3
Set Invisibility percent 100
```

Don't forget to delete your test objects!



DIFFERENTIATION

(15 minutes, optional)

- How would you change the quiz to have more answers or be longer?
- How could you change it to be a factual quiz—to change up whether the correct answer is A, B, or C?
- Turn the quiz into a survey that keeps track of how lots of people have answered—use this data to make a histogram.

REFLECTION

(5 minutes, optional)

- What is a variable?
- What is a conditional?
- How are variables and conditionals related or used together?
- How are events related to conditionals? How do you make a custom event? (When play button is tapped, repeat forever, check once if ...)
- What happens if there's a tie? How do we use conditionals to detect a tie? (if $a=b=c$)



GLOSSARY FOR OLDER STUDENTS

Ability/Function/Procedure/Subroutine: A saved set of blocks. What we call abilities in Hopscotch are known as functions or subroutines in other programming languages. Easily replicable routines are a key concept in computer programming, and allow you to scale your code and create complex programs.

Algorithm: Algorithms are at the heart of computer science; they are the recipes that computers follow to solve problems.

Bug: An error that a programmer has made in their code

Coding: Writing the rules of behavior for a computer to follow automatically; programming

Concurrency: Two or more things happening at the same time, or triggered by the same event

Conditional: Statements of the form "IF (something is true) THEN (do an action)"

Debugging: Finding mistakes in your code (bugs) and fixing them

Event: A trigger that the computer recognizes and causes it to do some action. In Hopscotch, events include "When the iPad is tapped" or "When the play button is tapped"

Iteration: the repetition of a process

Logic: the science of the formal processes of thinking and reasoning

Loop: a repeating set of instructions

Object: A character or text with its own rules on screen

Operator: a mathematical symbol that produces a value

Pair Programming: a technique in which two people work together on one device. Hopscotch is developed using pair programming :)

Program: a set of instructions a computer can understand

Programmer: a person who writes programs

Programming Language: a set of words, rules, blocks or instructions that can be used to write a program.

Random: Any number or item among a set. The lack of a pattern among items in a set.

Range: The highest and lowest number random can choose between

Rule: Rules tell your object what to do and when to do it. When you make an ability and pair it with an event, you create a rule.

Sequence: An ordered list of things (instructions, blocks, numbers, etc) which can be triggered by an event or repeated

Value: A holder for a number. Also known as a variable



GLOSSARY FOR YOUNGER STUDENTS

Ability: Code that can be reused

Algorithm: A recipe for a program

Coding: Telling computers what to do

Concurrence: Two things happening at the same time

Conditional: Statements of the form "IF (something is true) THEN (do an action)".

Debugging: Finding mistakes in your code and fixing them

Event: When something happens

Iteration: Having ideas and making mistakes, over and over

Logic: The process of making decisions

Loop: Code that repeats

Operator: A mathematical symbol that makes an equation

Program: A set of instructions a computer can understand

Programmer: A person who writes programs

Programming Language: A set of rules or blocks that can be used to write any program

Random: When there's no pattern

Range: The highest and lowest number random can choose between

Rule: Instructions that tell your computer what to do (the command) and when to do it (the event)

Sequence: The order in which instructions are given to the computer

Object: A character or text with its own rules

Value/Variable: A holder for a number

