JavaScript – theory and tasks

Loops

One of a computer's greatest abilities is to repeat a task over and over so we don't have to. Loops let us tell the computer to loop over a block of code so that we don't have to write out the same process over and over.

Loops are especially helpful when we have an array where we'd like to do something to each item in the array, like logging each item to the console.

There are two kinds of loops we will learn in this lesson:

1. for loops, which let us loop a block of code a known amount of times.
2. while loops, which let us loop a block of code an unknown amount of times.

Before we jump into writing a loop, let's write the result of a loop, so that we can better understand how loops work.

***Instructions***

*Write your code in the vacation.js file.*

1. Write an array and set it equal to a variable named vacationSpots. Inside of this array, list three places you'd like to visit.
2. Next, console.log each item in vacationSpots on a separate line. To do this, list out each item using property access.

Instead of writing out the same code over and over, let’s make the computer loop through our array for us. We can do this with for loops.

The syntax looks like this:

var animals = ["Grizzly Bear", "Sloth", "Sea Lion"];

for (var i = 0; i < animals.length; i++) {

console.log(animals[i]);

}

// Output:

// Grizzly Bear

// Sloth

// Sea Lion

Since this syntax is a little complicated, let's break it into 4 parts:

1. Within the for loop's parentheses, the *start condition* is var i = 0, which means the loop will start counting at 0.
2. The stop condition is i < animals.length, which means the loop will run as long as i is less than the length of the animals array. When i is greater than the length of the animals array, the loop will stop looping.
3. The *iterator* is i++. This means that each loop, i will have 1 added to it.
4. And finally, the code block is inside the { ... }. The block will run each loop, until the loop stops.

The secret to loops is that i, the variable we created inside the for loop's parentheses, is always equal to a number. To be more clear, the first loop, i will equal 0, the second loop, i will equal 1, and the third loop, i will equal 2.

This makes it possible to write animals[0], animals[1], animals[2] programmatically instead of by hand. We can write a for loop, and replace the hard coded number with the variable i, like this: animals[i].

***Instructions***

1. Let's replace your current code with a loop.

Write a for loop that loops through your vacationSpots array.

Inside the block of the for loop, use console.log to print each item in the vacationSpots array.

1. Now, add more to each item. Inside the code block, add some text to each item, like this:

console.log('I would love to visit ' + vacationSpots[i]);

If we can make a for loop run forwards through an array, can we make it run backwards through it? Of course!

We can make out loop run backwards by modifying the start, stop, and iterator conditions.

To do this, we'll need to edit the code between the for loop's parentheses:

1. The start condition should set i to the length of the array.
2. The stop condition should end when i is 0.
3. The iterator should subtract 1 each time, which is done with i--.

***Instructions***

1. We need to make three changes to our for loop:

Edit the start condition (var i = 0), to set i equal to the length of the vacationSpots array.

Then, set the stop condition ( i < vacationSpots.length) to stop when i is greater than or equal to 0.

Finally, change i++ to i-- to subtract 1 from i each loop.

1. Except we printed 'I would love to visit undefined'. Why did this happen?

It's because the length of vacationSpots is 3 and vacationSpots has items in its array at positions 0, 1, and 2.

Since JavaScript starts from 0, make the start condition the length of the vacationSpots array, minus 1.

Let's say that you and a friend would like to go on vacation together. You've both made arrays of your favorite places and you want to compare to see if any places match. This is a job for loops!

The big idea is that we can run a for loop inside another for loop to compare the items in two arrays.

Every time the outter for loop runs once, the inner for loop will run completely.

With two for loops, we can check to see if any of the your vacation spots match your friend's spots.

***Instructions***

*Write your code in the vacationCompare.js file.*

1. We are going to write this program from scratch. Start out by writing an variable named myPlaces and set it equal to an array with three places you'd like to visit.
2. Now, make another variable named friendPlaces and set it equal to an array with three places a friend might like to go.

Make sure that at least one of the places is the same as in your myPlaces array.

1. Write a for loop that iterates through each item in myPlaces and logs out each place.
2. You logged all of your places!

Now, inside of the existing for loop's block, write another for loop that loops over friendPlaces. This time, instead of using the i as the variable name, use j, so we don't overwrite any variables.

1. Look what printed to the console. Your first place printed, then all three of your friend's. Then your second place, then your friend's places again. And then one more time.

This is because the inner for loop runs completely every time the outer for loop runs once.

1. The purpose of the program is to see what you and your friend have in common. Let's utilize the === comparison with an if/else statement.

Inside the second for loop's block, write an if/else statement that checks if myPlaces[i] is equal to friendPlaces[j]. If it is, log to the console the place you have in common.

for loops are great, but they have a limitation: you have to know how many times you want the loop to run. What if you want a loop to run an unknown or variable number of times instead?

For example, if we have a deck of cards and we want to flip cards (loop a card flipping function) until we get a 'Spade', how could we write that in JavaScript?

That's the purpose of the while loop. It looks like similar to a for loop. Check it out:

while (condition) {

// code block that loops until condition is false

}

1. The loop begins with the keyword while
2. Inside the parentheses, we can insert a condition. As long as the variable evaluates to true the block of code will loop.
3. Inside the code block we can write any code we'd like to loop.

***Instructions***

*Write your code in the cards.js file.*

1. Let's write a program that flips cards until we get a 'Spade.' Start by creating a variable named cards and set it equal to this array:

['Diamond', 'Spade', 'Heart', 'Club']

1. Right under the array create a variable named currentCard and set it equal to 'Heart'.

This variable will hold the name of the card we just flipped. We are using 'Heart' as the first card.

1. Let's utilize a while loop to do two things:

* If the currentCard is not a 'Spade', then add a console.log to print the value of currentCard.
* Then, create a random number between 0 and 3 and put it in a variable named randomNumber.
* Then use the randomNumber to reassign currentCard to a new card from the cards array.

The while loop could like this:

while (currentCard !== 'Spade') {

console.log(currentCard);

var randomNumber = Math.floor(Math.random() \* 4);

currentCard = cards[randomNumber];

}

1. Outside the while loop, on the last line of the program, use console.log to log that the program found a spade.

Run the code a few times to see the output changing. You can see the while loop guessing a card, then seeing if it is a Spade, over and over, until it finds one.

In this unit we learned how to write less repetitive code with loops.

* for loops allow us to repeat a block of code a known amount of times.
* We can use a for loop inside another for loop to compare two lists.
* while loops are for looping over a code block an unknown amount of times.

At this point, you've learned the foundational concepts of JavaScript. Now we are going to apply them to the web with a JavaScript library called jQuery.