JavaScript – theory and tasks

Arrays

We’ve learned to do a number of interesting things with data using functions and using if/else statements. One thing that we haven't learned yet is how to organize and store data.

One way we organize data in real life is to make lists. Let's make one here:

*Bucket List:*

*0. Rappel into a cave*

*1. Take a falconry class*

*2. Learn to juggle*

Let's now write this list in JavaScript, as an array:

var bucketList = ['Rappel into a cave', 'Take a falconry class', 'Learn to juggle'];

Arrays are JavaScript's way of making lists. These lists can store different data types and they are ordered, meaning the position of each list item is numbered by JavaScript.

Let's start by making an array and then seeing what it can do throughout the rest of this lesson.

***Instructions***

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

1. Make a variable named bucketList, and set it equal to an array with three strings inside of it.
2. Use console.log to print bucketList to the screen.

Now, what if we want to select one item from an array?

Luckily, each item in an array has a numbered position. We can access an item using its number, just like we would in an ordinary list. There’s one catch though!

JavaScript counts starting from 0, not 1, so the first item in an array will be at position 0. This is because JavaScript is zero-indexed.

We can select the first item in an array like this:

var bucketList = ['Rappel into a cave', 'Take a falconry class', 'Learn to juggle'];

var listItem = bucketList[0];

console.log(listItem);

// Output: 'Rappel into a cave'

If we wanted the second item, we'd write:

var bucketList = ['Rappel into a cave', 'Take a falconry class', 'Learn to juggle'];

var listItem = bucketList[1];

console.log(listItem);

// Output: 'Take a falconry class'

***Instructions***

1. Create a variable named listItem and set it equal to the first item in your bucketList array.

Then use console.log to print the listItem variable to the console.

1. Now, set the listItem variable to the third item in the bucketList array.
2. Try to log the item at position [3] to the console.
3. Notice that you get undefined when you try to print position 3. That's because the array does not have an item at position 3, which is the fourth slot.

Fun fact: You can also access each individual character in a string the same way you do with arrays. For instance, you can write:

var hello = 'Hello World';

console.log(hello[6]);

// Output: W

'W' will be the output since it's the character in the 6th position. This works because JavaScript internally stores strings in a similar way that it stores arrays.

It is often convenient to know how many items are inside of an array.

We can find this out by using one of an array's built in properties, called .length. We can attach this to any variable holding an array and it will return the number of items inside.

As an example:

var bucketList = ['Rappel into a cave', 'Take a falconry class'];

console.log(bucketList.length);

// Output: 2

***Instructions***

1. Find the .length of your bucketList array and log it to the console.
2. Fun fact: .length is also a property for strings.

For instance, you can write 'Hello World'.length, and it will output 11 (the number of characters in "Hello World").

JavaScript has a surprise for us: it has built in functions for arrays that help us do common tasks! Let's learn two of them.

First, push() allows us to add items to the end of an array. Here is an example of how this is used:

var bucketList = ['item 0', 'item 1', 'item 2'];

bucketList.push('item 3', 'item 4');

The method push() would make the bucketList array look like:

['item 0', 'item 1', 'item 2', 'item 3', 'item 4'];

Check out how push() works here:

It connects to bucketList with a period.

Then we call it like a function. That's because push() is a function and one that JavaScript allows us to use right on an array.

Connecting a function like this is common in JavaScript. Think: we've been connecting .log to console this whole time!

***Instructions***

1. Add two more items to your bucketList array using push.
2. Now, use console.log to print your bucketList array to make sure your items were added.

Now that we can push() items into an array, let's pop one off, using pop().

pop() is similar to push(), except that it deletes the last item of an array. Here's an example:

var bucketList = ['item 0', 'item 1', 'item 2'];

bucketList.pop();

console.log(bucketList);

// Output: [ 'item 0', 'item 1' ]

Notice that 'item 2' was deleted from the end.

***Instructions***

1. Use the pop method to delete the last element from your array.
2. Log bucketList to the console to make sure it worked.
3. push and pop are just two of many methods we have for arrays. You can learn more array methods here.

In this lesson, we learned these concepts regarding arrays:

* Arrays are lists and are a way to store data in JavaScript. Each item inside of an array is at a numbered position. Arrays are created with brackets [].
* We can access one item in an array using it's numbered position, with syntax like: myArray[0].
* Arrays have a length property, which allows you to see how many items are in an array.
* Arrays also have their own methods, including push and pop, which add and subtract items from an array, respectively.

In the next lesson we'll learn how to loop over our arrays, which can bring our lists to life!