**Arrays**

A foundational concept of programming is how to organize and store data.

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

New Year's Resolutions: 1. Rappel into a cave 2. Take a falconry class 3. Learn to juggle

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

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

Arrays are JavaScript's way of making lists. These lists can store any data types (including strings, numbers, and booleans) and they are ordered, meaning each item has a numbered position.

**Property Access**

Each item in an array has a numbered position. We can access individual items using their numbers, just like we would in an ordinary list.

Something important to note is that JavaScript starts counting from 0 rather than 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:

let newYearsResolutions = ['Rappel into a cave', 'Take a falconry class', 'Learn to juggle']; console.log(newYearsResolutions[0]); // Output: 'Rappel into a cave'

You can also access individual characters in a string. For instance, you can write:

let 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 stores strings in a similar way that it stores arrays.

**Instructions**

**1.**

Individual elements of arrays can also be stored to variables.

Create a variable named listItem and set it equal to the first item in your newYearsResolutions array using square bracket notation ([]).

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

Hint

The first item within an array is always at position [0].

**2.**

Now, console.log() the third item in the newYearsResolutions array without using a variable.

Hint

console.log(array[index])

**3.**

Try to log the item at position [3] to the console. What is logged to the console?

|  |
| --- |
| let newYearsResolutions = ['a','b','c'];  console.log(newYearsResolutions);  let listItem = newYearsResolutions[0];  console.log(listItem);  console.log(newYearsResolutions[2]);  console.log(newYearsResolutions[3]); |
| [ 'a', 'b', 'c' ]  a  c  undefined |

# Update Elements

In the previous exercise, you learned how to access elements of an array or a string using their index number. You can also change elements of an array using their indices.

let seasons = ["Winter", "Spring", "Summer", "Fall"]; seasons[3] = "Autumn"; console.log(seasons) //Output: //Winter //Spring //Summer //Autumn

In the example above, the seasons array contained the names of the four seasons.

However, we decided that we preferred to say "Autumn" instead of "Fall".

seasons[3] = "Autumn"; tells our program to change the item at index 3 of the seasonsarray to be "Autumn" instead of what is already there.

# length property

We may wish to know how many items are stored 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.

let newYearsResolutions = ['Rappel into a cave', 'Take a falconry class']; console.log(newYearsResolutions.length); // Output: 2

In the example above, we log newYearsResolutions.length to the console. This code retrieves the length property of the newYearsResolutions array and logs it to the console. This array has two items, so 2 would be logged to the console.

# push Method

JavaScript has built in methods for arrays that help us do common tasks. Let's learn some of them.

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

let newYearsResolutions = ['item 0', 'item 1', 'item 2']; newYearsResolutions.push('item 3', 'item 4');

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

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

How does .push() work?

1. It connects to newYearsResolutions with a period.
2. 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.

Another array method, .pop(), is similar to .push(). This method removes the last item of an array.

let newYearsResolutions = ['item 0', 'item 1', 'item 2']; newYearsResolutions.pop(); console.log(newYearsResolutions); // Output: [ 'item 0', 'item 1' ]

In the example above, calling .pop() on the newYearsResolutions array removed item 2 from the end.

# More Array Methods

There are many more array methods than just .push() and .pop(). You can read about all of the array methods that exist on the [Mozilla Developer Network (MDN)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array)documentation.

.pop() and .push() modify the array on which they're called. However, there are some array methods that don't modify the array. Be sure to check MDN to understand the behavior of the method you are using.

Some methods that JavaScript developers use frequently are .join(), .slice(), .splice(), .shift(), .unshift(), and .concat() amongst many others.

Below, we will explore some methods that we have not learned yet. We will use these methods to edit a grocery list. As you complete the steps, you can consult the MDN documentation to learn what each method does!

|  |
| --- |
| https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array |

Use the .shift() method to remove the first item from the array groceryList.

Log the new groceryList to the console.

Hint

Read the .shift() documentation on [MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/shift).

Your code will look similar to this:

array.method(); console.log(array);

|  |
| --- |
| let groceryList = ['orange juice', 'bananas', 'coffee beans', 'brown rice', 'pasta', 'coconut oil', 'plantains'];  groceryList.shift();  console.log(groceryList); |
| [ 'bananas',  'coffee beans',  'brown rice',  'pasta',  'coconut oil',  'plantains' ] |

Use the .unshift() method to add 'popcorn' to the beginning of your grocery list.

Log the new grocery list to the console.

You may wish to delete the console.log() statement from the previous step.

Hint

Read the .unshift() documentation on [MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift).

Your code will look similar to this:

array.method(argument); console.log(array);

|  |
| --- |
| let groceryList = ['orange juice', 'bananas', 'coffee beans', 'brown rice', 'pasta', 'coconut oil', 'plantains'];  groceryList.shift();  groceryList.unshift('popcorn');  console.log(groceryList); |
| [ 'popcorn',  'bananas',  'coffee beans',  'brown rice',  'pasta',  'coconut oil',  'plantains' ] |

**3.**

You're in a hurry so you decide to ask a friend to help you with your grocery shopping. You want him to pick up the bananas, coffee beans, and brown rice.

Use .slice() to provide him with a list of these three things.

Log this part of the list to the console. Unlike the two previous checkpoints, you should do both of these steps in one line.

Hint

Read the .slice() documentation on [MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice).

Your code will look similar to this:

console.log(array.method(first, last+1));

|  |
| --- |
| let groceryList = ['orange juice', 'bananas', 'coffee beans', 'brown rice', 'pasta', 'coconut oil', 'plantains'];  groceryList.shift();  groceryList.unshift('popcorn');  console.log(groceryList);  console.log(groceryList.slice(1,4)); |
| [ 'popcorn',  'bananas',  'coffee beans',  'brown rice',  'pasta',  'coconut oil',  'plantains' ]  [ 'bananas', 'coffee beans', 'brown rice' ] |

Log the grocery list to the console one more time.

Notice that the groceryList array still contains the same items it had in Step 2.

|  |
| --- |
| let groceryList = ['orange juice', 'bananas', 'coffee beans', 'brown rice', 'pasta', 'coconut oil', 'plantains'];  groceryList.shift();  groceryList.unshift('popcorn');  console.log(groceryList);  console.log(groceryList.slice(1,4));  console.log(groceryList); |
| [ 'popcorn',  'bananas',  'coffee beans',  'brown rice',  'pasta',  'coconut oil',  'plantains' ]  [ 'bananas', 'coffee beans', 'brown rice' ]  [ 'popcorn',  'bananas',  'coffee beans',  'brown rice',  'pasta',  'coconut oil',  'plantains' ] |

# Arrays with let and const

You may recall that you can declare variables with both the let and const keywords. Variables declared with let can be reassigned.

Variables that are assigned with const cannot be reassigned. However, arrays that are declared with const remain mutable, or changeable.

This means that we can change the contents of an array, but cannot reassign the variable name to a new array or other data type.

The instructions below will illustrate this more clearly. Pay close attention to the similarities and differences between the condiments array and the utensils array as you complete the steps.

**Instructions**

**1.**

Below the two existing arrays, add your favorite condiment to the condimentsarray using .push().

Log the updated array to the console.

|  |
| --- |
| let condiments = ['Ketchup', 'Mustard', 'Soy Sauce', 'Sriracha'];  const utensils = ['Fork', 'Knife', 'Chopsticks', 'Spork'];  condiments.push('condi');  console.log(condiments); |
| [ 'Ketchup', 'Mustard', 'Soy Sauce', 'Sriracha', 'condi' ] |

Below your code from Step 1, reassign condiments to be an array that contains a single string.

Log the result to the console.

Hint

condiments = ['a string'];

|  |
| --- |
| let condiments = ['Ketchup', 'Mustard', 'Soy Sauce', 'Sriracha'];  const utensils = ['Fork', 'Knife', 'Chopsticks', 'Spork'];  condiments.push('condi');  console.log(condiments);  condiments = ['re-condi'];  console.log(condiments); |
| [ 'Ketchup', 'Mustard', 'Soy Sauce', 'Sriracha', 'condi' ]  [ 're-condi' ] |

Below your code from Step 2, remove the last item from the utensils array using .pop().

Log the updated array to the console.

|  |
| --- |
| let condiments = ['Ketchup', 'Mustard', 'Soy Sauce', 'Sriracha'];  const utensils = ['Fork', 'Knife', 'Chopsticks', 'Spork'];  condiments.push('condi');  console.log(condiments);  condiments = ['re-condi'];  console.log(condiments);  utensils.pop();  console.log(utensils); |
| [ 'Ketchup', 'Mustard', 'Soy Sauce', 'Sriracha', 'condi' ]  [ 're-condi' ]  [ 'Fork', 'Knife', 'Chopsticks' ] |

Below your code from Step 3, reassign the utensils array to be a new array with only your favorite utensil inside of it.

You should have received an error!

|  |
| --- |
| let condiments = ['Ketchup', 'Mustard', 'Soy Sauce', 'Sriracha'];  const utensils = ['Fork', 'Knife', 'Chopsticks', 'Spork'];  condiments.push('condi');  console.log(condiments);  condiments = ['re-condi'];  console.log(condiments);  utensils.pop();  console.log(utensils);  utensils = ['re-uten']; |

**Review Arrays**

Nice work! In this lesson, we learned these concepts regarding arrays:

* Arrays are lists and are a way to store data in JavaScript.
* Arrays are created with brackets [].
* Each item inside of an array is at a numbered position, starting at 0.
* We can access one item in an array using its numbered position, with syntax like: myArray[0].
* We can also change an item in an array using its numbered position, with syntax like myArray[0] = "new string";
* Arrays have a length property, which allows you to see how many items are in an array.
* Arrays have their own methods, including .push() and .pop(), which add and remove items from an array, respectively.
* Arrays have many other methods that perform different functions, such as .slice() and .shift(). You can read the documentation for any array method on the [Mozilla Developer Network](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array) website.
* Variables that contain arrays can be declared with let or const. Even when declared with const, arrays are still mutable; they can be changed. However, a variable declared with const cannot be reassigned.