Section 10: ES6 Maps

After completing this section you will:

- [ ] How to Create a Map
- [ ] Adding Data to a Map with Set()
- [ ] Using the Map() Constructor
- [ ] Understand How to Loop through a Map
- [ ] Extract Keys and Values from a Map

Introduction

So far you've learned two ways to hold data in memory in JavaScript. Variables can hold single data points of various types. Arrays can contain lists of data of a single type. You've also seen a number of examples of how these simple data structures are used in coding. Unfortunately, life is not so simple, and real-life data can be complex and layered. That's why we have **data structures**, which are objects that hold complex forms of data. Imagine, for example, you had a tournament bracket that you needed to store in a program. It would be challenging to store this data in a variable or simple array.

In this section, we're going to look at a new data structure in ES6 called the Map.

Maps with New

Maps are essentially sets of *key/value* pairs. The key is designed to describe the value. For example, if you had a baseball team and you wanted to store player position and name, they position would be your key, and the player name would be your value.

Here's an example of map data:

Position (Key)	Name (Value)
President	Trump
Vice-President	Pence
Secretary of State	Pompeo
Speaker of the House	Pelosi

In this particular example, the keys and values are all strings. In all Maps your keys must all be of a consistent type, and your values must all be of a uniform type.

In JavaScript we'd declare our Map like this:

```
<div id="output"></div>
<script>
    let team = new Map();

</script>
```

In this case, `team` will be the identifier for our Map object. This Map object is empty at this point and needs to be populated with data.

Using Set()

The Set() method is used to set key-value pairs in a Map object. Set is used like this:

```
team.set('pitcher', 'CC Sabathia');
```

The name of the set object is team, and the set function has two arguments. The first argument is the key and the second argument is the value.

Let's set() some key/value pairs in our Map.

```
<div id="output"></div>
<script>
    let team = new Map();
    team.set('pitcher', 'CC Sabathia');
    team.set ('catcher', 'Gary Sanchez');
    team.set ('centerfield', 'Brett Gardener');
    team.set ('shortstop', 'Didi Gregorius');
</script>
```

> Even though we've set a number of values in our map, they don't have a particular order. The order in which they are set doesn't matter as the items in the Map can only be recalled using the key or value.

It is also common to chain set commands one after another. This is a bit of shorthand and isn't functionally different from discretely writing each set() command individually. The example below chains set() commands:

```
let goodSongs = new Map();
    bands.set('Journey', 'Faithfully')
    .set('REO', 'Keep on Rolling')
    .set('Scandal', 'The Warrior')
    .set('The Cure', 'Pictures of you')
    .set('Rolling Stones', 'Angie')
    .set('Mr. Mister', 'Kyrie');
```

> Note: The carriage returns in the previous example are to enhance readability but are not required.

The Map() Constructor

In the fine JavaScript tradition of being able to do everything two or three different ways, you can also declare maps using the class constructor. Again, there is no functional difference, and this is simply a preference.

The following Map is created using the class constructor:

```
let family = new Map([
      ['mother', 'Joan'],
      ['father', 'Rick'],
      ['brother', 'Brett'],
      ['sister-in-law', 'Kerry'],
      ['cousin', 'Joyce']
  ]);
```

If you use this method, be a bit careful with the brackets as there are multiple sets and it's easy to leave one out or add an extra causing a syntax error.

For... Of Loops

As with most data structures, the most convenient way to get all the data out of a Map is with a loop. The `for...of` loop is specially designed for this.

The for...of loop will iterate through all of the key/value pairs in your map. Let's use the family map we created above and loop through it.

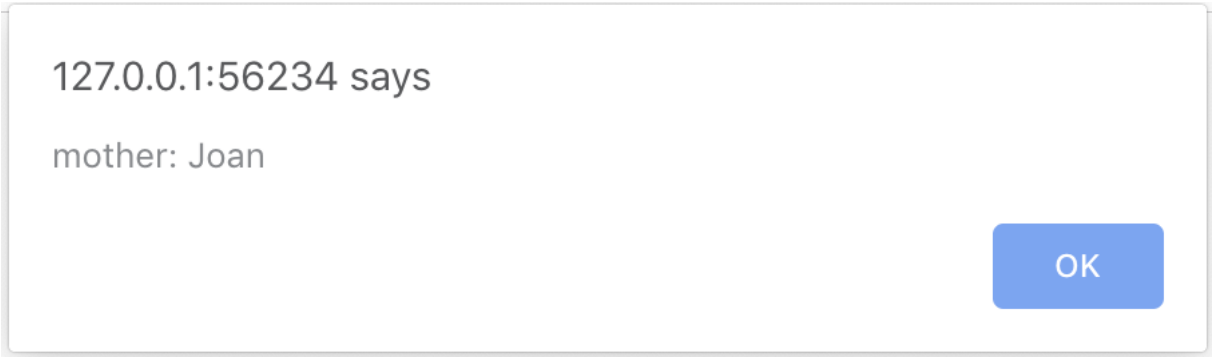
```
for(let r of family.entries())
       {
           alert(r);
       }
```

This example will loop through the Map assigning each key/value pair to the variable `r` in each iteration. In this case, each pair will be displayed in an alert box.

The `entries()` method is key here as it iterates through the loop one entry at a time returning both the key and value. These entries are returned as an array with the key in position 0 and the value in position 1. With this in mind, we could change the above code like this:

```
for(let r of family.entries())
        {
            alert(r[0] + ": " + r[1]);
        }
```

Now each member of the Map would be displayed like this:



Extracting Keys

It is equally easy using the `for...is` loop to extract the keys from the Map object by using the `keys()` method.

```
for(let r of family.keys())
        {
            alert(r);
        }
```

As the loop iterates through the Map object, the key values are assigned to the variable `r` in each iteration.

Extracting Values

Replacing `keys()` with `values()` allows you to iterate the individual values in the Map object.

```
for(let r of family.values())
        {
            alert(r);
        }
```

The loop above would alert() out the values Joan, Rick, etc. from the original Map.

Debug This:

There are two errors in the code below preventing it from functioning correctly. When the code is working

correctly, the Add to Map button adds the values entered into the fields by the user to the map. The second button displays the map in the empty `div` with the id `result`.

Remember there are only two errors here! Good luck.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Map Exercise</title>
    </head>
    <body>
        <h1>Add to the Map</h1>
        <label for="name">Name</label>
        <input type="text" id="name" /><br/>
        <label for="numericalGrade">Grade</label>
        <input type="number" min="0" max="100" id="numericalGrade"/><br/>
        <button id="btnToAddToMap">Add to Map Object</button>
        <button id="btnDisplayMap">Display Map</button>
        <div id="result"></div>

        <script>
            const ADDBUTTON = document.getElementById('btnToAddToMap');
            const DISPLAYBUTTON = document.getElementById('btnDisplayMap');
            let grades = new Map();


            ADDBUTTON.onclick = function(){
                const name = document.getElementById('name');
                const grade = document.getElementById('numericalGrade');
                grades.add(name.value, grade.value);
                name.value = "";
                grade.value = "";

            };

            DISPLAYBUTTON.onclick = function(){
                let out= "";
                for(let x of grades.entries())
                    {
                        out += x[0] + ": "  x[1];
                        out += "<br/>";
                    }
                document.getElementById('result').innerHTML = out;
            }
        </script>
    </body>
</html>
```

Submit This: My To-Dos

It's clean up time at Camp Whack-a-Doo. The problem is the campers never know which chores are theirs. You have been tasked with creating a JavaScript program to store the campers and assign tasks. The data should be stored in a map.

When the program starts the user should be instructed to enter a campers name, and then a task. The tasks should be limited to the following:

Front Room Sweep
Back Room Sweep
Outside Grounds
Clean Toilet
Clean Sink
Trash Patrol and Supplies

The campers' names and assignments should be stored in a map with the task being the key and the camper's name the value.

The interface should include a button that, when clicked lists all the campers names and their clean-up assignments.

> The requirements here are purposefully a bit vague giving you some latitude on how you implement a correct solution. Just make sure you meet the requirements listed.

For this course visit https://www.dropbox.com/request/Nl7bSaAe11ZI0PgLRonA to submit your assignments.