Exercise 04 – Collections

Objectives

In this exercise, you will investigate JavaScript arrays and their functions.

Overview

This exercise is broken down into 3 separate sections, allowing you to explore collections and objects in JavaScript.

This exercise is scheduled to run for **30 mins (10 for each part)**.

Exercise Set Up

1. In VSCode, open the file called **CollectionsAndObjects.html** from the **Starter** folder inside **Ex04** of the **Exercises** folder.

You will note that there are commented areas provided for each section of the exercise.

2. If you have rebooted your computer or stopped *live-server* from running since the last exercise, then use VSCode's integrated terminal to navigate to the **Exercises** folder and run the command **live-server.**
3. Open your browser (if it doesn't automatically) and navigate to **http://localhost:8080** and follow the path to open **Ex04→Starter→ Collections.html.**

Part 1 – Creating and Managing Arrays

In this part of the exercise, you will practice creating an array, manipulating it and producing an output from it.

4. Under the comment for Part 1, declare an array called **quote** that contains four *strings*, "**I**", "**am**" "**your**" and "**friend**".
5. Log the array to the console.
6. Save the file and observe the browser to check the output - *you should see details of an array and expanding it will show the values and their indexes*.
7. Access the index of the array that contains the string "your" and log the array element to the console.

```
console.log(quote[2]);
```

8. Save the file and observe the browser to check the output.
9. Using the **pop** function, remove the string "**friend**" from the end of the array.
10. Using the **push** function, add the string "**father**" to the end of the array.
11. Log the array to the console again.
12. Save the file and observe the browser to check the output.
13. Use the **unshift** function to add the string "**Luke**" to the start of the array.
14. Log the array to the console again.
15. Save the file and observe the browser to check the output.

16. Use the **toString** function inside a console.log to output the array as a string
17. Save the file and observe the browser to check the output.

There are two things wrong with the output. The first is that it the string is concatenated by commas and the second is that the 'quote' is actually a misquote! We're going to generate an output in a different way by looping through the array and creating a new string. We're going to fix the misquote by detecting the erroneous word in the array and replacing it with the correct word! Let's do that first.

To do this, we are going to detect if indeed the erroneous word is in the array. If it is, we are going to find the index that the word is at and then use this information to replace that index with the correct word.

18. Declare a variable called erroneousWord and set it to a string with the misquoted word from the array (it's Luke if you didn't know!).
19. Set a variable called lukeIsHere using the find() function to see if the quote array contains the erroneousWord. The code is:

```
let lukeIsHere = quote.find(n => { return n === erroneousWord});
```

The syntax inside the find function will feel a little alien at the moment but go with it as it is explained later in the course.

20. Declare a variable called lukeIsAt without assigning it.
21. If lukeIsHere has been set to true, find the index the erroneousWord sits at using the findIndex() function and set lukeIsAt to the value of the index. The code is:

```
let lukeIsAt = quote.findIndex(n => { return n ===
erroneousWord});
```

22. Still inside the if block, use the value of lukeIsAt to set that index in the quote array to the string "No"
23. Log out the array and ensure that the expected result is outputted in the browser.

To sort out the display of the quote, we need to create a new string then loop through the array, adding a space into the string after each word apart from the first word, to which we'll append a comma and a space and the final word to which we will append an exclamation mark.

24. Declare a variable called output and set it to be an empty string.
25. Create a for loop that:
    a) Loops through the quote array.
    b) Executes when the loop counter is less than the length of the array.
    c) Adds an exclamation mark to the output string, if we are at the last element in the array.
    d) Adds a comma and a space to the output string, if the current element is 'No'.
    e) Otherwise adds a space to the output string.
26. Log out the output string.
27. Save the file and then check your browser output to ensure that the correct quote is displayed.

No, I am your father!

Part 2 – Maps

This part of the exercise is to explore Maps (there is no part on Sets as the methods are pretty much the same, just without allowing duplicates!). To do this we will create a set of key value pairs that hold information.

Set Up

1. In the file CollectionsAndObjects.html, comment out the first block of script tags and add another set.
2. Ensure that live-server is still outputting your file and check that the output console is clear.

Maps

For clarity of instructions, the steps to save and observe the browser have been omitted after each instruction that affects the output.

3. Create a new **Map** object called **hanSolo** and add the following *key/value* pairs to it:
   a) **vehicle** - **Millenium Falcon**;
   b) **bff** - **Chebacca**;
   c) **sweetheart** - **Leia**.
4. Access the properties that you have just declared by logging out the following details:
   a) The **size** of the map **hanSolo**;
   b) Han Solo's **vehicle** name (***HINT*** - use the **Map.get()** method);
   c) If Han Solo has a **sweetheart** (***HINT***: use the **Map.has()** method);
   d) If Han Solo is a (***has***) **Jedi**.
5. Add another *key/value* pair to **hanSolo** that sets a key **son** to **Ben** and log this new property to the console.
6. Iterate over **hanSolo** using a **for...of** loop that uses both the *key* and the *value* of each pair, logging out each pair.
7. Manipulate the map by:
   a) *Changing* the value of **bff** to **Luke** and log out **hanSolo**;
   b) *Deleting* the key/value pair **son** and log out **hanSolo**;
   c) *Clearing* the Map and logging it out.

Part 3 – Objects

This part of the exercise is to explore Objects. The previous part, whilst completely valid code is probably not the best way to store details about an individual thing. The object construct is much more appropriate, much less typing and easier to access and manipulate!

Set Up

1. In the file **CollectionsAndObjects.html**, comment out the last block of script tags used in Part 2 and add another set.

2. Ensure that live-server is still outputting your file and check that the output console is clear.

Objects

3. Create a new **Object** called **darthVader** and add the following *key/value* pairs to it:
   a) **allegiance** - **Empire**;
   b) **weapon** - **lightsabre**;
   c) **sith** - **true** (boolean value).
4. Access the properties that you have just declared by logging out the following details:
   a) DarthVader's **allegiance**;
   b) Darth Vader's **weapon**;
   c) If Darth Vader is a **sith**;
   d) The value of **Jedi** from Darth Vader;
   e) The number of properties Darth Vader has (see the line of code below for this)

```
console.log(Object.keys(darthVader).length);
```

Quick explanation - Object.keys is a function that takes an object and returns an array of the keys in it. By appending .length to it, we return the number of keys in the object.

5. Add *key/value* pairs to **darthVader** that:
   a) Sets a key of **children** to **2**;
   b) Sets a key of **childNames** to the array **['Luke', 'Leia']**;
   And then log the **children** property and the *value of the first element* in the **childNames** array.
6. Iterate over **darthVader** using a **for...in** loop that uses both the *key* and the *value* of each pair, logging out each pair.
7. Manipulate the object by:
   a) *Changing* the value of **allegiance** to **The light side** and log out **darthVader**;
   b) *Deleting* the key/value pair **children** and log out **darthVader**;
   Hint: use the code below:

```
delete darthVader.children;
```

   c) *Clearing* the object and logging it out.