Quick Lab 6 - JavaScript Functions

Objectives

* To investigate JavaScript functions.

Activity

1. In VSCode, create the file index.js and grab the functions.js file from my gist**.**
2. Create a **script** tag that imports **functions.js**.
3. Open **index.html** with the live server

The browser window will open, but there will be nothing to view!

Part 1 – Defining and using functions and understanding scope

In this part of the exercise, you will practice writing a function to perform some actions and investigate the scope of the variables within it. In addition, you will look at the scope of variables. You will notice that the script already contains an array of objects that contain some details about different films.

1. Under the comment for Part 1, declare a function called findMovie that takes an argument called movieTitle.
2. In the body of the function create a for…of loop of the movies array where:
   1. The loop body should:
      1. Check to see if the current movie title is the same as the movieTitle passed into the function and if it is, log out details of the movie in a suitable string;
      2. Log out the value of movie before the loop's closing brace;
   2. The value of movie should be logged before the function closes.
3. Call the findMovie function with an argument of "Star Wars".
4. Log out the value of movie.
5. At this point, save your file and check the output.

The expected outcome is that there is a Reference Error - but which console.log is, or console.logs are, causing it/them?

1. Comment out the offending console.log(s) and check your output.

You should see all 5 movies logged, with the string you wrote for a found movie being outputted before the movie object for it is logged itself.

Two of the console.log statements added produced a Reference Error. This is because of the scope of the variable movie. As it is declared as part of the for…of loop, its scope is limited to inside the body of this block (i.e. between the { } that immediately follows the for). As long as execution remains inside this loop, the variable movie is in scope.

Once the loop finishes and execution returns to the level above (i.e. back to the body of the function) and the variable movie is no longer in scope and therefore referring to it in the code causes the Reference Error. It follows that if it is not available here, it will also not be available after the line that calls the function has completed execution, again causing a Reference Error.

Note: Because movieTitle is part of the function block, it is accessible throughout the execution of the function, including inside any blocks that are used within the function body (i.e. in the for and if blocks).

Note: Because the const movies is declared at script level (i.e. inside the script tag) and at the top of it, it is available to all blocks of code that live inside this script tag.

1. Under the last line, define a variable called movie set to the value of   
   "Thor: Ragnorok".
2. Uncomment the console.logs of movies and add another log of the value of movie under the declaration of the variable from step 11.
3. Observe the results.

You will notice that the same Reference Errors as before are present.

1. Change the declaration of movie to have the var keyword in front of it (rather than let) and make sure that all console.logs are uncommented.
2. Observe the results.

What you should see this time is two undefined values. The differences are all to do with concepts called hoisting and 'temporal dead zones'. More details of which can be found, with a good explanation of let at:

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let>

1. Add a call to the findMovie with the argument set to movie, saving and observe the output.

There is no output as there is no movie in the movies array with the title   
"Thor: Ragnorok" and therefore the loop completes without ever entering the if condition.

Part 2 – Creating functions that return data

Functions rarely just execute code and then the program continues. It is more usual that a function will manipulate some data and then return some data which can then be used further. This part of the exercise will allow you to experiment with returning data from a function.

1. Comment out the section of code for Part 1, leaving the movies array intact and work under the comment for Part 2.
2. Declare a function called returnMovie that takes movieTitle as an argument and has a function body that:
   1. Uses a for…of loop on the movies array with a loop body that:
      1. Checks to see if the title property of the current movie matches the movieTitle supplied to the function;
      2. If it does, it should simply return the current movie;
      3. Logs out the current value of movie;
   2. Logs out "Any text, any text at all".
3. In the body of the script, declare a variable called myMovie and set it to the result of calling returnMovie with an argument of "Avengers: Infinity War".
4. Log out the value of myMovie, save and observe the output.

If you have created your returnMovie function you should observe the following:

* Each of the movies that appear BEFORE the selected movie are logged out as the loop has executed for each of these movies
* The movies that are AFTER the selected movie are not logged out because the presence of the return statement stops the execution of the loop and indeed the function (so that "Any text, any text at all" is also not shown)
  + - The execution 'returns' to its call point with the value of whatever is returned

1. Access the properties of myMovie to produce and log a string as a sentence with them in it, saving and observing your output.

What happens if we try to pass a movie title that doesn't exist in the movies array into returnMovie? Let's find out!

1. Declare a variable myOtherMovie and set its value to a call to returnMovie with an argument of "Thor: Ragnorok".
2. Log out the value of myOtherMovie and observe the output.

The first thing that we notice is that the whole of the movies array has been logged out and the text "Any text, any text at all". This is because the title was not found, and the function completed its execution fully and never returned a value…or did it?

The next thing that we notice is that the console.log of myOtherMovie has outputted undefined. It looks like we've never set the value of myOtherMovie because we haven't! Let's fix that…

1. Comment out the logging of "Any text…" and add a line that returns the string `Movie not found`.
2. Save and observe the output.

The logging of myOtherMovie now outputs "Movie not found".

The code is still not very reusable as if I want to log out the details of a movie, I have to supply the string inside a console.log. Also, what happens if it is already a string (because it is a movie not in the array)? Our output would be very messy! Step up another function!

1. Create a function called myMovieDetails that takes a variable anyMovie as an argument.
2. Check that the typeof anyMovie is an 'object' and return a suitable string if it is and simply return anyMovie if it isn't.
3. Inside a console.log, call myMovieDetails with an argument of myOtherMovie.
4. Observe the results.

It should output: Movie not found.

Can we use a function as the argument to another function? Yes we can!

1. Repeat the last instruction instead passing in returnMovie with an argument of "Jaws" as the argument to the myMovieDetails function.
2. Observe the results.

It should output the details for Jaws in your defined string.

This is the end of Quick Lab 6