Quick Lab 1 - JavaScript Types

Objectives

* To understand how types work in JavaScript.

Activity

1. In VSCode, create the files **index.js** and **index.html** in a new folder.
2. Add a script tag to **index.html** that references **index.js**.
3. In index.js, add code to declare a number and a console.log to output it:

let numTest = 45.324568;

console.log(numTest);

1. Open your index.html using the live server.
2. When the browser opens, view the console and check that the number is displayed.

Press F12 to see the developer tools and choose the Console tab. You should see the value of the numTest variable displayed.

You have created a Number data type object in the program stack, this is a 64bit number. Next, we will explore the Number type a little more with some of its methods.

We will create a new variable called twoDecimalPoints and use a method to truncate the number.

1. Under the last code you wrote in index.js, add the following code:

let twoDecimalPoints = numTest.toFixed(2);

console.log(twoDecimalPoints);

1. Save the file and your browser should automatically refresh to display the value of twoDecimalPoints as 45.32. Notice that this number is now BLACK not BLUE? The toFixed function converts the number to a string!
2. Under your last line of code in index.js, create a stringTest variable, as shown in the code segment below.

(Please be sure to add the text exactly as it appears; otherwise, the notes will not match up to what you will see in the console.)

let stringTest = `I am the very model of a modern major general`;

let indexOfM = stringTest.indexOf(`m`);

console.log(indexOfM);

1. Save the code and observer the browser console.

You will see a value of 3, examine the string and you will see that the m is the fourth character, so there are three characters before the first m.

1. Change the m within the indexOf method call to a capital M, save and observe the output in the browser again.

This time, the console.log will return a -1 value. The -1 value is telling us that there is no match within the string at all proving that string searches are case sensitive.

What if we convert the string to upper case?

1. Before the indexOfM line, add the following code:

stringTest = stringTest.toUpperCase();

1. Save and observe the output in the browser again.

The output will, once again, give a value of 3. Behind the scenes, the string is an indexed collection of characters and the search function is making its way through the letters character by character until it makes a match.

With that concept in mind, we will use the principals to learn how to slice a string.

1. Add the following code under the last line, then capture start and end in a console.log, save and observe the output.

let start = stringTest.indexOf("MODEL");

let end = stringTest.lastIndexOf('MAJOR');

This time, we have matched based upon words, but you could search for file paths or extensions; for instance, if we were reading from a form.

The two integer values held can be used to create a substring from the longer one using string's substring method.

1. Add the following lines of code to the end of your code, save and observe the output.

let subStr = stringTest.substring(start, end);

console.log(subStr);

The console should now return a value of "MODEL OF A MODERN".

Let's finish up this exercise by writing this content to the browser window using the document.write method (we will look at the document object in more depth later on).

1. Add the following lines of code to the end of your code, save and observe the output.

document.write("<p>" + subStr + "</p>");

We have used an operator here: the + sign which we have used to concatenate the string together and mix our string value with some hard-coded HTML to create new content to the page. With that done, let’s learn some more about operators.

This is the end of Quick Lab 1