Strings

Recall that a String is a variable-type that stores a sequence of characters. Javascript includes several built-in functions for working with strings. The following is a list of some string functions available in JS. Your task is to investigate the functions and experiment with each function in order to understand it. Create a file demonstrating the use of each function.

In Javascript, every character in a String can be referred to with an *index*, with the first character having an index of 0. **For example**, in the String "ICS20", the character 'C' would have an index of 1.

In addition, all String functions ***return*** a new string and **do not** modify the original variable. Formally, we then say that Strings are ***immutable***, meaning they cannot be changed directly through a function.

A complete list of available String functions can be found at www.w3schools.com.

substring(start, end) or substring(start)

⇒ returns the string starting from the index *start* and ending one index before *end*. If *end* is not included, the substring will include all characters in the String from *start*.

```
Ex:     var title = "I love this class";
        var part = title.substring(2,6);        //part has the value 'love'
```

Also, notice that to find the substring, we had to call the function using the String title. This will be used with almost all String functions.

indexOf(search, index) or indexOf(search)

⇒ returns the starting position of the *first occurrence* of the substring *search*. If the substring is not found, -1 is returned. The parameter *index* is the position to start searching. If *index* is not included, searching automatically starts at index 0.

```
Ex:     var word = "brighten";
        var search = "right";
        var result = word.indexOf(search);    //result has the value 1
```

lastIndexOf(search)

⇒ returns the starting position of the *last occurrence* of the substring *search*. Works similarly to indexOf() above.

replace(find, replacement)

⇒ returns a new String with the first occurrence of *find* replaced by *replacement*.

```
Ex:     var word = "Microsoft";
        var replacement = "HA";
        var result = word.replace("o", replacement);        //result has the value 'MicrHAsoft'
```

toLowerCase() or toUpperCase()

⇒ returns the string argument as an all lowercase or uppercase string.

```
Ex:     var original = "brighten";
        var nowUpper = original.toUpperCase();
```

length

⇒ the length property returns the number of characters in a string.

includes(search)

⇒ returns true if the given String includes the substring search. Returns false otherwise.

startsWith(search) or endsWith(search)

⇒ returns true if the given String begins are ends with the substring *search*. Returns false otherwise.

charAt(index)

⇒ returns the character located at a certain index of a String.

charCodeAt(index)

⇒ returns the ASCII code corresponding to a character at a certain *index*.

Ex: var word = ” super” ;
var character = word.charCodeAt(0) ; //character has the value 115

115 is the decimal value of the character ‘s’ in the ASCII table. Check for a link to the ASCII table on the website.

String.fromCharCode(integer)

⇒ returns the character corresponding to the *integer* between (0-255) using the ASCII table.

Ex: var result = String.fromCharCode(115) //result has the value ‘s’

Notice that use the variable name String when calling this function, as opposed to the name of a existing variable.

String Comparisons (>, <, ==, !=, >=, <=)

⇒ You are able to use comparison operators to compare strings. Javascript compares the ASCII codes of the starting character in each String.

⇒ String comparisons can be tricky because comparisons using operators like <, >, == are case sensitive since they are using the ASCII code to compare them. Ex: the comparison “Apple” == “apple” is false because the uppercase “A” and lowercase “a” are represented by two different ASCII codes.