ACCOLITE UNIVERSITY, JULY 2020

Javascript

Assignment Submission

By: Ayush Malik

1. Implementation of all array functions on an array.

```
var arr = ['a','b','c']
var arr2 = ['d','e','f']
var arr1 = [1,2,3,4,5]
var arr4 = ["z","y","x"]
```

concat()

```
arr
(3) ["a", "b", "c"]
arr2
(3) ["d", "e", "f"]
arr.concat(arr2);
(6) ["a", "b", "c", "d", "e", "f"]
```

every()

```
arr1 = [1,2,3,4,5]
(5) [1, 2, 3, 4, 5]
arr1.every((currentValue) => currentValue < 4);
false
```

filter()

```
arr1 = [1,2,3,4,5]
(5) [1, 2, 3, 4, 5]
arr1.filter((currentValue) => currentValue < 4);
(3) [1, 2, 3]
```

forEach()

```
arr1 = [1,2,3,4,5]
(5) [1, 2, 3, 4, 5]
arr1.forEach(i => console.log(i));
1
2
```

```
3
4
5
```

indexOf()

```
arr
(3) ["a", "b", "c"]
arr.indexOf("c")
2
```

join()

```
arr
(3) ["a", "b", "c"]
arr.join()
"a,b,c"
```

lastIndexOf()

```
arr
(3) ["a", "b", "c"]
arr.indexOf("f")
-1
```

map()

```
arr1 = [1,2,3,4,5]
(5) [1, 2, 3, 4, 5]
arr1.map(function(item) {return ++item;});
(5) [2, 3, 4, 5, 6]
```

pop()

```
arr
(3) ["a", "b", "c"]
arr.pop()
"c"
arr
(2) ["a", "b"]
```

push()

```
arr
(2) ["a", "b"]
arr.push("c")
3
arr
(3) ["a", "b", "c"]
```

reduce()

```
arr1 = [1,2,3,4,5]
(5) [1, 2, 3, 4, 5]
var temp=arr1.reduce(function(acc,curr){return acc+curr;});
console.log(temp);
15
```

reduceRight()

```
arr1 = [1,2,3,4,5]
(5) [1, 2, 3, 4, 5]
var temp=arr1.reduce(function(acc,curr){return acc-curr;});
console.log(temp);
-13
```

reverse()

```
arr
(3) ["a", "b", "c"]
arr.reverse()
(3) ["c", "b", "a"]
```

shift()

```
arr
(3) ["a", "b", "c"]
arr.unshift("d");
4
arr;
(4) ["d", "a", "b", "c"]
```

slice()

```
arr
(3) ["a", "b", "c"]
arr2
(3) ["d", "e", "f"]
arr.concat(arr2)
(6) ["a", "b", "c", "d", "e", "f"]
arr.concat(arr2).slice(3,5);
(2) ["d", "e"]
arr.concat(arr2);
(6) ["a", "b", "c", "d", "e", "f"]
```

some()

```
arr1 = [1,2,3,4,5]
(5) [1, 2, 3, 4, 5]
arr1.some((currentValue) => currentValue < 4);
true
```

toSource()

```
Function is obsolete and no longer supported by chrome.
```

sort()

```
arr4=["z","y","x"]
(3) ["z", "y", "x"]
arr4.sort()
(3) ["x", "y", "z"]
```

splice()

```
concat_array
(6) ["a", "b", "c", "d", "e", "f"]
concat_array.splice(2,4);
(4) ["c", "d", "e", "f"]
concat_array
(2) ["a", "b"]
```

toString()

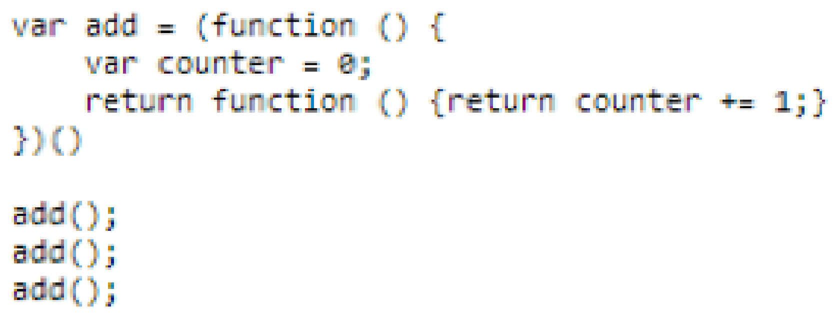
```
arr1=[1,2,3,4,5]
(5) [1, 2, 3, 4, 5]
arr1.toString()
"1,2,3,4,5"
```

unshift()

```
arr;
(4) ["d", "a", "b", "c"]
arr.shift();
"d"

arr;
(3) ["a", "b", "c"]
```

2. Explanation of the working of the following code:



```
var add = (function () {
    var counter = 0;
    return function () {return counter += 1;}
})()

add();
add();
add();
```

This function is used to implement a counter. For our understanding, let us rewrite this code as follows:

```
var add=(function_outside(){
var counter=0;
return function_inside(){ return counter+=1;}
})()
add();
add();
add();
```

Note: functions have been named like this just for explanation.
Function_outside is only called once and it sets counter value to 0.
Now, in the variable ‘add’ is assigned function_inside return by function_outside.
So, everytime add() is called only function_inside is called.
All these calls share the global variable counter and hence the above code gives the output:

```
1
2
3
```

This is similar (but not exactly same) as writing the code as:

```
var counter=0;
var add=function(){ return counter+=1;}
}
add();
add();
add();
```

This is done to ensure counter values are shared across function calls to add(). If the counter function had been defined as below:

```
var add=function(){
var counter=0;
return counter+=1;}
```

```
}
add();
add();
add();
```

Everytime the add() function would have been called, counter would have been re-set to 0, thus returning the output:

```
1
1
1
```

3. Differences between \n and \r.

The differences between \n and \r have been specified below:

- \n is called the new line character. \r stands for carriage return.
- Different operating systems handle them differently. Windows uses '\n\r' to signify that the enter key was presses. Linux and modern Mac OS use '\n' to convey the same. Classic Mac OS used \'r' for new line.
- For C and most languages '\n' is the new line character, which is further translated for OS.
- For electromechanical type writers, \r indicates the carriage to go back leftwards until it reaches the leftmost point. \n commands the paper roller to roll up hence going to the next line.
- \r\n is the standard line-termination for text formats on the Internet.
- The reason \r is followed by \n in the notation of Windows and on the Internet is that it draws inspiration from a traditional typewriter where first the carriage is reset (\r) and then the paper is rolled (\n).
- These days generally \r and \n act similarly in character-mode terminals

4. Web page implementation for performing the following operation:

- **Checking for the specified regular expression**
- **Sort, filter >10, increment and display of 1-4 elements on a hard coded array**

The required codes are attached in **regex.html** and **array_manipulations.html** respectively.