JavaScript Assignment

Submitted By – Ayushi Garg

Question 1 –

index.html | sol.js

sol.js > ...

```javascript
var array = [1,2,3,4,5,6,7,8,9,10];

console.log(array.concat([11,12]));

console.log(array.every((item)=>{
    return item*1;
}));

console.log(array.filter((item)=>{
    if(item%2==0)
    return item;
}));

array.forEach((item)=> item*1);

console.log(array.indexOf(4));

console.log(array.join());

console.log(array.lastIndexOf(7));

console.log(array.map((item)=>{
    return item*10;
}));

console.log(array.pop());

console.log(array.push(11,12,13));

console.log(array.reduce((first,second)=>{return second - first;}));

console.log(array.reduceRight((first,second)=>{return second - first;}));

console.log(array.reverse());

console.log(array.shift());

console.log(array.slice(3));
```

```
console.log(array.some((item)=>{return item>6}));

console.log(array.sort());

console.log(array.splice(3,4));

console.log(array.toString());

console.log(array.unshift(-2,-1,0));
```

```
Sources  Elements  Console  Network  Performance  Memory  Application
top  Filter  Default levels
▶ (12) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]          sol.js:58
true                                                    sol.js:60
▶ (5) [2, 4, 6, 8, 10]                                  sol.js:64
3                                                       sol.js:71
1,2,3,4,5,6,7,8,9,10                                    sol.js:73
6                                                       sol.js:75
▶ (10) [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]        sol.js:77
10                                                      sol.js:81
12                                                      sol.js:83
7                                                       sol.js:85
-7                                                      sol.js:87
▶ (12) [13, 12, 11, 9, 8, 7, 6, 5, 4, 3, 2, 1]          sol.js:89
13                                                      sol.js:91
▶ (8) [8, 7, 6, 5, 4, 3, 2, 1]                          sol.js:93
true                                                    sol.js:95
▶ (11) [1, 11, 12, 2, 3, 4, 5, 6, 7, 8, 9]              sol.js:97
▶ (4) [2, 3, 4, 5]                                      sol.js:99
1,11,12,6,7,8,9                                         sol.js:101
10                                                      sol.js:103
>
```

Question 2 –

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

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

In the above piece of code, it can be seen that add is a function that returns a self-invoking function. It has a variable named counter, assigned 0 as an initial value. The self-invoking function increments the value of counter by 1 and return the same to the calling statement.

As soon as the control reaches first add (), the add function is called and self-invoking function increments counter value to 1 and return it to the add function which results in returning of 1 to the called add (). As the counter belongs to local scope of add function, the value of incremented counter stays for the next call. Hence the output of the above code is as follows -

```
1
2
3
>
```

Question 3 –

```
var func = function(str){
    var reg1 = /^lion/;
    var reg2 = /cat$/;
    var reg3 = /ab+c/;
    if(reg1.exec(str)){
        console.log("True and the location is 0");
    }
    else if(reg2.exec(str)){
        console.log("True"+" and the location is "+ (str.length - 3));
    }
    else if(reg3.exec(str)){
        console.log("True and the location is " + reg3.exec(str).index);
    }
    else{
        console.log("False");
    }
}

func("lion is the fastest");
func("I am strong as lion");
func("I love my cat");
func("My cat is white in color");
func("I am going to tabbcen to buy a lion");
```

```
True and the location is 0
False
True and the location is 10
False
True and the location is 15
>
```

Question 4 –

index.html | sol.js

sol.js > ...

```javascript
var func = function(array){
    array.sort(function(first, second){return first - second});
    console.log(array);
    for(var i=0;i<array.length;i++){
        array[i] = array[i]*10;
    }
    console.log(array);
    console.log(array.filter((item)=>{
        if(item%3==0)
        return item;
    }));
}
var numbers = [1,3,5,7,9,2,4,6,8];
var list = [23,56,45,67,24,12,4,78,30];
func(numbers);
func(list);
```

Sources | Elements | Console | Network | Performance | Memory | Application | Security | Lighthouse

top | Filter | Default levels

```
(9) [1, 2, 3, 4, 5, 6, 7, 8, 9]
(9) [10, 20, 30, 40, 50, 60, 70, 80, 90]
(3) [30, 60, 90]
(9) [4, 12, 23, 24, 30, 45, 56, 67, 78]
(9) [40, 120, 230, 240, 300, 450, 560, 670, 780]
(5) [120, 240, 300, 450, 780]
>
```

Question 5 –

== : Let's say we take a string and a number variable. When we compare the two variable using == operator, the operator automatically converts one type to other and return true if the value of the variables is same else it returns false.

=== : The operator will check the type of both variables as well the values. If both type and values of the variables are same, then it will return true. Otherwise, it returns false.

For example –

2 == 2 // True

2 == '2' // True

2 === 2 // True

2 === '2' // False