**W2D1 - Assignment 7-8 | Scopes, Closures Practice**

**Name:Owais Ur Rehman Awan**

**Student ID: 611121**

**Q1. Determine what this JavaScript code will print out?**

x= 1;

var a = 5;

var b = 10;

var c = function(a, b, c) {

document.write(x);

document.write(a);

var f = function(a, b, c) {

b = a;

document.write(b);

b = c;

var x = 5;

}

f(a,b,c);

document.write(b);

}

c(8,9,10);

document.write(b);

document.write(x);

}

Answer: undefined 8 8 9 10 1

Explanation: c(8,9,10) is called first and it prints, x undefined , a 8 , and then it calls f(8,9,10) and inside this function 8 is assigned to b and prints b then in next step c is assigned to b, but remember then its inside the function f scope. After executing f , it prints 9 for b. At last it prints b 10 and x 1 as initialized.

**Q2. Define Global Scope and local Scope:**

**Global Scope:**

Any variable declared outside the function blocks belongs to the global scope and can be accessible from anywhere like

Var x=1;

Const x=Function(){

Var a=x \* x;

}

Here x has the global scope and can be accessed from anywhere.

Local Scope:

Any variable declared inside certain block and which have limited access is knows as local storage. Like

Var x=1;

Const x=Function(){

Var a=0;

}

Now here Var a has the local storage which is only accessible inside the function.

**Q3. Determine access for scope in below code?**

//Scope A

function xFunc() {

//Scope B

function xFunc() {

//Scope C

};

};

a) Statement in scope A have access to variables defined in Scope B and C. **FALSE**

b) Statement in scope B have access to variables defined in Scope A. **TRUE**

c) Statement in scope B have access to variables defined in Scope C. **FALSE**

d) Statement in scope C have access to variables defined in Scope A. **TRUE**

e) Statements in scope C have access to variables defined in Scope B. **TRUE**

Explanation: Scope A is accessible to both B and C. B is accessible to C. C is only accessible to itself.

**Q4. What will be printed by the following code?**

var x=9;

function myFunction(){

return x\*x;

}

document.write(myFunction());

x=5;

document.write(myFunction());

Answer: 8125

Explanation: Initially x has value 9 so myFunction returns 81 and later it changes to 5 so myFunction returns 25.

**Q5. What will the alert print out in the following?**

var foo=1;

function bar(){

if(!foo){

var foo=10;

}

alert(foo);

}

bar();

Answer: Alert with value 10.

Explanation: !foo has value true so 10 is assigned to it.

**Q6: Solution:**

const x = (function() {

const count = {

counter: 0,

add: function() {

return this.counter += 1;

},

reset: function() {

return this.counter = 0;

}

}

console.log(count.add());

console.log(count.add());

console.log(count.reset());

})();

**Q7: Solution:**

Free variable are variable that are neither local variable nor passed as a parameter. In the context of closure method

Var count is the free variable as its not local as well as not passed as parameter.

**Q8: Solution:**

const make\_adder = function(x) {

let counter = 0

return function() {

counter += x;

}

}

const add5 = make\_adder(5);

add5(); add5(); add5();

**Q 9: Solution:**

If we can simply put all our code in IIFE function , then the name from the global namespace has been removed now and scope changes from global scope to functional scope.

**Q 10: Solution:**

const Employee = (function() {

let name = "";

let age = "";

let salary = "";

function setSalary(salary) {

this.salary = salary;

}

function setage(age) {

this.age = age;

}

function setName(name){

this.name=name;

}

function getName() {

return this.name;

}

function getAge() {

return this.age;

}

function getSalary() {

return this.salary;

}

function IncreaseSalary(percentage) {

this.setSalary(this.getSalary + (this.getSalary \* percentage) / 100);

}

function incrementAge() {

this.setage(this.getAge + 1);

}

//Public Members

return {

setAge: setage,

setSalary: setSalary,

setName: setName,

getName: getName,

getAge: getAge,

getSalary: getSalary,

IncreaseSalary: IncreaseSalary,

incrementAge: incrementAge

}

})();

Employee.setAge(30);

Employee.setName("owais");

Employee.setSalary(11000);

console.log(`Employee with Name: ${Employee.getName()} with Salary: ${Employee.getSalary()} and Age: ${Employee.getAge()}`);

**Q11: Solution:**

Employee.address=””;

Employee.setAddress=function(adress){

This.address=address;

}

Employee.getAddress=function(){

return this.address;

}