Javascript Module Exercises

1. Determine what this Javascript code will print out (without running it):

```
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);
var x = 10;
 }
c(8,9,10);
document.write(b);
document.write(x);
}
```

Answer :

Undefined 8 8 9 10

2. Define Global Scope and Local Scope in Javascript.

Answer :

Global variables can be used (and changed) by all scripts on the page (and in the window). Global variables live as long as your application (your window / your web page) lives.

A local variable can only be used inside the function where it is defined. It is hidden from other functions and other scripting code. Local variables have short lives. They are created when the function is invoked and deleted when the function is finished

3. Consider the following structure of Javascript code:

```
// Scope A
function XFunc () {
// Scope B
function YFunc () {
// Scope C
};
};
```

(a) Do statements in Scope A have access to variables defined in Scope B and C?

No, They do not have access to variables

(b) Do statements in Scope B have access to variables defined in Scope A?

Yes B can access variables defined in Scope A

(c) Do statements in Scope B have access to variables defined in Scope C?

No, it does not have access to C

(d) Do statements in Scope C have access to variables defined in Scope A?

Yes C can access variables defined in Scope A

(e) Do statements in Scope C have access to variables defined in Scope B?

Yes Yes B can access variables defined in Scope B

4. What will be printed by the following (answer without running it)?

```
var x = 9;
function myFunction() {
return x * x;
}
document.write(myFunction());
x = 5;
document.write(myFunction());
```

Answer

81 25

5.

```
var foo = 1;
function bar() {
if (!foo) {
var foo = 10;
}
alert(foo);
}
bar();
```

What will the alert print out? (Answer without running the code. Remember ‘hoisting’.)?

Answer

10

6. Consider the following definition of an add( ) function to increment a counter variable:

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

Modify the above module to define a count object with two methods: add( ) and reset( ). The count.add( ) method adds one to the counter (as above). The count.reset( ) method sets the counter to 0.

Answer

```
var object = (function () {
var counter = 0;
 return{
 add :function () {
 counter += 1;
  alert(counter);
},
reset:function () {
 counter =0;
}
};
 })();
```

7. In the definition of add( ) shown in question 6, identify the "free" variable. In the context of a function closure, what is a "free" variable?

Answer

The free variable for question 6 is counter.

8. The add( ) function defined in question 6 always adds 1 to the counter each time it is called. Write a definition of a function make_adder(inc), whose return value is an add function with increment value inc (instead of 1).

Here is an example of using this function: add5 = make_adder(5); add5( ); add5( ); add5( ); // final counter value is 15 add7 = make_adder(7); add7( ); add7( ); add7( ); // final counter value is 21

Answer

```
var object = (function () {
var counter = 0;
 return{
 make_adder:function(value){
 counter+=value;
 alert(counter);
 },
 add :function () {
 counter += 1;
  alert(counter);
},
reset:function () {
 counter =0;
}
};
 })();
```

9. Suppose you are given a file of Javascript code containing a list of many function and variable declarations. All of these function and variable names will be added to the Global Javascript namespace. What simple modification to the Javascript file can remove all the names from the Global namespace?

Answer
We can use closure. It makes it possible for a function to have private variables and methods.

10. Using the Revealing Module Pattern, write a Javascript definition of a Module that creates an Employee Object with the following fields and methods:

Private Field: name
Private Field: age
Private Field: salary
Public Method: setAge(newAge)
Public Method: setSalary(newSalary)
Public Method: setName(newName)
Private Method: getAge( )
Private Method: getSalary( )
Private Method: getName( )
Public Method: increaseSalary(percentage) // uses private getSalary( )
Public Method: incrementAge( ) // uses private getAge( )

Answer

```
var Employee  = (function () {
let name ;
let age ;
let salary ;

let  setAge=function (newAge){
          this.age=newAge;
} ;
let setSalary=function (newSalary){
          this.salary=newSalary;

};
let setName=function (newName){
this.name=newName;

};
let getAge=function(){

            return age;
};

let getSalary=function (){

            return age ;
};

let getName=function (){

            return name;
};

let increasesSalary=function (percentage){

            this.salary=salary*percentage;

};

let incrementAge =function (){

            age+=1;
};

return {
setAge:setAge,
setSalary:setSalary,
setName:setName,
getAge:getAge,
getName:getName,
incrementAge:incrementAge,
increasesSalary:increasesSalary

}

})();
```

11. Rewrite your answer to Question 10 using the Anonymous Object Literal Return Pattern.

Answer
```
var Employee  = (function () {
let name ;
let age ;
let salary ;
return {
        setAge:function (newAge){
                this.age=newAge;},
        setSalary:function (newSalary){
                this.salary=newSalary;},
        setName:function (newName){
                this.name=newName;},
        getAge:function(){
                return age; },
        getSalary:function (){
                return age ; },
        getName:function (){
                return name;},
        increasesSalary:function (percentage){
                this.salary=salary*percentage;},
        incrementAge:function (){
                age+=1;}
}
})();
```

12. Rewrite your answer to Question 10 using the Locally Scoped Object Literal Pattern.

Answer :
```
        var Employee  = (function () {
        let name ;
        let age ;
        let salary ;
        let object={};
          object.setAge=function (newAge){
                this.age=newAge;} ;
         object.setSalary=function (newSalary){
                this.salary=newSalary;};
        object.setName=function (newName){
        this.name=newName;};
        object.getAge=function(){
                return age; };
        object.getSalary=function (){
                return age ;
```

```
            };
            object.getName=function (){
                        return name; };
            object.increasesSalary=function (percentage){
             this.salary=salary*percentage;};
            object.incrementAge =function (){
                        age+=1;
            };

            return object;

            })();
```

13. Write a few Javascript instructions to extend the Module of Question 10 to have a public address field and public methods setAddress(newAddress) and getAddress( ).

Answer

```
var address;
var Employee  = (function () {
let name ;
let age ;
let salary ;
let object={};
  object.setAge=function (newAge){
            this.age=newAge;} ;
 object.setSalary=function (newSalary){
            this.salary=newSalary;};
object.setName=function (newName){
this.name=newName;};
object.getAge=function(){
            return age; };
object.getSalary=function (){
            return age ;
};
object.getName=function (){
            return name; };
object.increasesSalary=function (percentage){
 this.salary=salary*percentage;};
object.incrementAge =function (){
            age+=1;
};

return object;
})();



Employee.setAddress=function(newaddress){
		this.address=newaddress;
}

Employee.getAddress=function (){
		return address;
}
```

14. What is the output of the following code?

```
const promise = new Promise((resolve, reject) => {
reject("Hattori");
});
promise.then(val => alert("Success: "+ val))
.catch(e => alert("Error: " + e));
```

Answer :
Hattori

15. What is the output of the following code?

```
const promise = new Promise((resolve, reject) => {
resolve("Hattori");
setTimeout(()=> reject("Yoshi"), 500);
});
promise.then(val => alert("Success: " + val))
 .catch(e => alert("Error: " + e));
```

Answer :
Hattori

16. What is the output of the following code?

```
function job(state) {
 return new Promise(function(resolve, reject) {
 if (state) {
 resolve('success');
 } else {
 reject('error');
 }
 });
}
let promise = job(true);
promise.then(function(data) {
 console.log(data);
 return job(false);})
 .catch(function(error) {
 console.log(error);
 return 'Error caught';
});
```

Answer :
Success
error