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, 1

2. Define Global Scope and Local Scope in Javascript.

Global Scope : a scope which defines the global environment for function and variables
Local Scope: the inner scope defined by a function to determine the visibility and accessibility of variables

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

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

Answer: B,D,E

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

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

Answer: 10

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

```
var count  = (function() {
            var counter = 2;
            var add = function() {
            return counter += 1;
            }
            var reset = function(){
            return counter;
            }


            return{
```

```
            add:add
        }

        return{
            reset:reset
        }


    })();
```

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:

COUNTER is the free variable

Free variable are those variables which are not locally defined inside the closure function or not passed as a parameter to the closure but can be used by the closure.

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:

```
9   function make_adder(inc){
10              var counter = 0;
11              var add = function() {
12              return counter += inc;
13              }
14
15              return add;
16          }
17
18          add5 = make_adder(5);
19          console.log(add5( )); console.log(add5( ));  console.log(add5(
    ));  // final counter value is 15
20          add7 = make_adder(7);
21          console.log( add7()); console.log( add7()); console.log( add7());
    // final counter value is 21
```

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:
The simplest modification to do would be to use module pattern, wrapping the the entire code construction inside an anonymous function.

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

```
var Employee = function(){

            var name;//private field
            var age;//private field
            var salary;//private field

            //public method
            function setAge(newAge){
                age = newAge;
            }
            //public method
            function setSalary(newSalary){
                salary = newSalary;
            }
            //public method
            function setName(newName)
            {
                name - newName;
            }
            //private method
            function getAge( )
            {
                return age;
            }
            //priavte method
            function getSalary()
            {
                return salary;
            }
            //private method
            function getName(){
                return name;
```

```
            }
            //public method
            function increaseSalary(percentage)
            {
                setSalary(getSalary + (getSalary * percentage));
            }

             // uses private getSalary( )
             //public method
            function  incrementAge( ){

                setAge(getAge()+=1);

            } // uses private getAge(

            return {
                setAge:setAge,
                setName:setName,
                setSalary:setSalary,
                increaseSalary:increaseSalary,
                incrementAge:incrementAge
            }

        }();
```

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

```
Var Employee = function(){
            var name;//private field
            var age;//private field
            var salary;//private field
            //private method
            function getAge( )
            {
                return age;
            }
            //priavte method
            function getSalary()
            {
                return salary;
            }
            //private method
            function getName(){
```

```
            return salary;
        }
        //another variation of returning: Anonymous object literal
        return {
            setAge : function(newAge){
                age = newAge;
            },
            setName : function(newName){
                name = newName;
            },
            setSalary : function(newSalary){
                salary = newSalary;
            },
            increaseSalary: function(percentage)
            {
                 setSalary(getSalary + (getSalary * percentage));
            },

            // uses private getSalary( )
            //public method
            incrementAge: function  ( ){

                setAge(getAge()+=1);
            } // uses private getAge(
        }

    }();
```

12.Rewrite your answer to Question 10 using the Locally scoped object literal Object Literal Pattern

```
13.     var Employee = function(){
14.
15.         var name;//private field
16.         var age;//private field
17.         var salary;//private field
18.
19.         let empObject = {};
20.
21.         //private method
22.         function getAge( )
23.         {
24.             return age;
25.         }
26.         //priavte method
```

```
27.             function getSalary()
28.             {
29.                 return salary;
30.             }
31.             //private method
32.             function getName(){
33.                 return name;
34.             }
35.
36.             empObject.setAge = function (){
37.                 age =  newAge;
38.             }
39.
40.             empObject.setName = function (){
41.                 age =  newName;
42.             }
43.             empObject.setSalary = function (){
44.                 age =  newSalary;
45.             }
46.
47.             empObject.increaseSalary = function (){
48.                 setSalary(getSalary + (getSalary * percentage));
49.             }
50.             empObject.incrementAge = function() {
51.
52.                 setAge(getAge()+=1);
53.
54.             }
55.       Return empObject;
56.         }();
```

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( ).

```
Employee.extension =  function (){
                var address;
                function setAddress(newAddress)
                {
                  address = newAddress;
                }

                function getAddress (){
```

```
            return address;
        }

        return{
            address:address,
            setAddress:setAddress,
            getAddress:getAddress
        }
    }
```

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 : Error: 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 : Success: 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