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);

}

Result: undefined 8 8 9 10 1

2. Define Global Scope and Local Scope in Javascript.

//Global Scope

function(){

//Local Scope

}

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?

Statements in Scope A can't access to Local Scope B and C.

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

Local statements in Scope B can access to variables defined in Scope A as free variable.

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

Statements in Scope B can't access to variables defined in Scope C. Because Scope C is local scope.

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

Statements in Scope C can access to variables defined in Scope A as free variable.

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

Statements in Scope C can't access to variables defined in Scope B. Because Scope B is local scope.

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());

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’.)?

Alert print 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.

var count = ( function (){

var add = function () {

var counter = 0;

return function () {

return counter += 1;

}

};

var reset = function(){

counter = 0;

return function(){

return counter;

};

};

return {

add: add(),

reset: reset()

};

})();

console.log(count.add());

console.log(count.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?

Free variable is used in Local scope from outer scope. In add() function counter is a free variable for returned function.

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:

var make\_adder = (function(inc){

let counter = 0;

return function(){

return counter += inc;

}

})

add5 = make\_adder(5);

add5( ); add5( ); add5( ); // final counter value is 15

add7 = make\_adder(7);

add7( ); add7( ); 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?

Module pattern encapsulates function and variables name from Global namespace.

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( )

var module = (function() {

let name;

let age;

let salary;

let setAge = function(age) {

this.age = age;

};

let setSalary = function(salary) {

this.salary = salary;

};

let setName = function(name) {

this.name = name;

};

let getAge = function() {

return this.age;

};

let publicGetAge = function() {

return getAge();

};

let getSalary = function() {

return this.salary;

};

let publicGetSalary = function() {

return getSalary();

};

let getName = function() {

return this.name;

};

let publicGetName = function() {

return getName();

};

let increaseSalary = function(percentage) {

return getSalary() + (getSalary \* 100) / percentage;

};

let incrementAge = function() {

return getAge() + 1;

};

return {

setAge: setAge,

setSalary: setSalary,

setName: setName,

increaseSalary: increaseSalary,

incrementAge: incrementAge,

publicGetAge: publicGetAge,

publicGetSalary: publicGetSalary,

publicGetName: publicGetName

};

})();

var employee = Object.create(module);

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

var module = (function() {

let name;

let age;

let salary;

let getAge = function() {

return this.age;

};

let getSalary = function() {

return this.salary;

};

let getName = function() {

return this.name;

};

return {

setAge: function(arg\_age) {

age = arg\_age;

},

setSalary: function(arg\_salary) {

salary = arg\_salary;

},

setName: function(arg\_name) {

name = arg\_name;

},

increaseSalary: function(percentage) {

return getSalary() + (getSalary \* 100) / percentage;

},

incrementAge: function() {

return getAge() + 1;

},

publicGetAge: function() {

return getAge();

},

publicGetSalary: function() {

return getSalary();

},

publicGetName: publicGetName = function() {

return getName();

}

};

})();

var employee = Object.create(module);

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

var module = (function() {

let name;

let age;

let salary;

let getAge = function() {

return age;

};

let getSalary = function() {

return salary;

};

let getName = function() {

return name;

};

let employee\_methods = {

setAge: function(arg\_age) {

age = arg\_age;

},

setSalary: function(arg\_salary) {

salary = arg\_salary;

},

setName: function(arg\_name) {

name = arg\_name;

},

increaseSalary: function(percentage) {

return getSalary() + (getSalary \* 100) / percentage;

},

incrementAge: function() {

return getAge() + 1;

},

publicGetAge: function() {

return getAge();

},

publicGetSalary: function() {

return getSalary();

},

publicGetName: publicGetName = function() {

return getName();

}

}

return employee\_methods;

})();

var employee = Object.create(module);

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( ).

Add public address field, setAddress and getAddress functions as a property to Module object.

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));

Hattori print in alert.

15. What is the output of the following code?

const promise = new Promise((resolve, reject) => {

reject(“Hattori”);

setTimeout(()=> reject(“Yoshi”), 500);

});

promise.then(val => alert(“Success: “ + val))

.catch(e => alert(“Error: “ + e));

First it will print "Hattori" in alert and after 500 millseconds print "Yoshi" in alert.