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); //output is 1

document.write(a); //output is 8

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

b = a;

document.write(b); //output is 8

b = c;

var x = 5;

}

f(a,b,c);

document.write(b); //output is 9

var x = 10;

}

c(8,9,10);

document.write(b); //output is 10

document.write(x); //output is 1

}

1 8 8 9 10 1

2. Define Global Scope and Local Scope in Javascript.

There is one global scope, and each function defined has its own (nested) local scope.

If we declare a variable, it is defined globally.

If we define a function and create variables inside it, those variables are locally scoped.

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

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

Yes

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

No

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

Yes

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

Yes

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

var x = 9;

function myFunction() {

return x \* x;

}

document.write(myFunction());//output is 81

x = 5;

document.write(myFunction());;//output is 25

81 25

5.

var foo = 1;

function bar() {

//but if alert(foo); is called here before assign, it will get undefined output.

if (!foo) {

var foo = 10;

}

alert(foo);//output is 10

}

bar();

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

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 counter = 0;

return{

add: function(){

counter += 1;

return counter;

},

reset : function(){

counter = 0;

return counter;

},

}

})();

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?

‘counter’ variable is the free variable.

The free variable is a variable referred to by a function that is not one of its parameters or local variables.

A closure occurs when a(n inner) function is defined and it attaches itself to the free variables from the surrounding environment to "close" up those stray references.

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

var make\_adder = function(num){

var counter = 0;

return function(){

return counter += num;

};

};

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?

Writing all codes in a module.

(function() {

your code...

})();

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 Employee = function() {

let name;

let age;

let salary;

let setName = function(newName) {

name = newName;

};

let setAge = function(newAge) {

age = newAge;

};

let setSalary = function(newSalary) {

salary = newSalary;

};

let getName = function() {

return name;

};

let getAge = function() {

return age;

};

let getSalary = function() {

return salary;

};

let increaseAge = function() {

age = parseInt(getAge()) + 1;

};

let increaseSalary = function(percentage) {

salary = parseFloat(getSalary()) + (parseFloat(getSalary()) \* (parseFloat(percentage) / 100));

};

let showData = function() {

alert(getName() + "-" + getAge() + "-" + getSalary());

};

return {

setName: setName,

setAge: setAge,

setSalary: setSalary,

increaseAge: increaseAge,

increaseSalary: increaseSalary,

showData: showData

};

};

var emp = new Employee();

emp.setName("Win");

emp.setAge("24");

emp.setSalary(100);

emp.showData();

emp.increaseAge();

emp.increaseSalary(90);

emp.showData();

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

var Employee = function() {

let name;

let age;

let salary;

let getName = function() {

return name;

};

let getAge = function() {

return age;

};

let getSalary = function() {

return salary;

};

return {

setName: function(newName) {

name = newName;

},

setAge: function(newAge) {

age = newAge;

},

setSalary: function(newSalary) {

salary = newSalary;

},

increaseAge: function() {

age = parseInt(getAge()) + 1;

},

increaseSalary: function(percentage) {

salary = parseFloat(getSalary()) + (parseFloat(getSalary()) \* (parseFloat(percentage) / 100));

},

showData: function() {

alert(getName() + "-" + getAge() + "-" + getSalary());

}

};

};

var emp = new Employee();

emp.setName("Win");

emp.setAge("24");

emp.setSalary(100);

emp.showData();

emp.increaseAge();

emp.increaseSalary(90);

emp.showData();

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

var Employee = function() {

let myObject = {};

let name;

let age;

let salary;

let getName = function() {

return name;

};

let getAge = function() {

return age;

};

let getSalary = function() {

return salary;

};

myObject.setName = function(newName) {

name = newName;

};

myObject.setAge = function(newAge) {

age = newAge;

};

myObject.setSalary = function(newSalary) {

salary = newSalary;

};

myObject.increaseAge = function() {

age = parseInt(getAge()) + 1;

};

myObject.increaseSalary = function(percentage) {

salary = parseFloat(getSalary()) + (parseFloat(getSalary()) \* (parseFloat(percentage) / 100));

};

myObject.showData = function() {

alert(getName() + "-" + getAge() + "-" + getSalary());

};

return myObject;

};

var emp = new Employee();

emp.setName("Win");

emp.setAge("24");

emp.setSalary(100);

emp.showData();

emp.increaseAge();

emp.increaseSalary(90);

emp.showData();

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( ).

var Employee = (function() {

let myObject = {};

let name;

let age;

let salary;

let getName = function() {

return name;

};

let getAge = function() {

return age;

};

let getSalary = function() {

return salary;

};

myObject.setName = function(newName) {

name = newName;

};

myObject.setAge = function(newAge) {

age = newAge;

};

myObject.setSalary = function(newSalary) {

salary = newSalary;

};

myObject.increaseAge = function() {

age = parseInt(getAge()) + 1;

};

myObject.increaseSalary = function(percentage) {

salary = parseFloat(getSalary()) + (parseFloat(getSalary()) \* (parseFloat(percentage) / 100));

};

myObject.showData = function() {

alert(getName() + "-" + getAge() + "-" + getSalary());

};

return myObject;

})();

var address;

Employee.setAddress = function(newAddress) {

address = newAddress;

};

Employee.getAddress = function() {

return address;

};

Employee.setName("Win");

Employee.setAge("24");

Employee.setSalary(100);

Employee.showData();

Employee.increaseAge();

Employee.increaseSalary(90);

Employee.showData();

Employee.setAddress("Fairfield, Iowa");

alert(Employee.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));

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));

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';

});

success

error