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);

}

Undefined889101

1. Define Global Scope and Local Scope in Javascript.

Global scope: refers to variable defined outside of a function and is accessible from any file.

Local scope: any variable declared inside a function block is referred as a variable with local scope.

1. Consider the following structure of Javascript code:

// Scope A

function XFunc () {

// Scope B

function YFunc () {

// Scope C

};

};

* + 1. Do statements in Scope A have access to variables defined in Scope B and C? No
    2. Do statements in Scope B have access to variables defined in Scope A? Yes
    3. Do statements in Scope B have access to variables defined in Scope C? No
    4. Do statements in Scope C have access to variables defined in Scope A? Yes
    5. Do statements in Scope C have access to variables defined in Scope B? Yes

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

var x = 9;

function myFunction() {

return x\*x;

}

document.write(myFunction());//81 will be printed

x = 5;

document.write(myFunction());//25 will be printed

.

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’.)?

foo will be hoisted but remain undefined. Output will be 10.

1. 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(){

return counter += 1;

}

reset:function(){

return counter = 0;

}

}

})();

1. 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 is a free variable in the context of a function closure.

1. 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).

var make\_adder = function (inc) {

var counter = 0;

return function() {

return counter += inc;

}

};

add5 = make\_adder(7);

add5();

add5();

print(add5());//15

1. 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?

Putting all the names within module function will remove them from the Global namespace.

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

Employee = (function(){

var name;

var age;

var salary;

function getAge(){

return age;

}

function getSalary(){

return age;

}

function getName(){

return age;

}

setAge: setAge(newAge){

age = newAge;

}

setSalary: setSalary(newSalar){

age = newSalary;

}

setName: setName(newName){

age = newName;

}

incSalary:increaseSalary(percentage){

salary = getSalary() + getSalary\*percentage;

}

incAge:incrementAge(){

Age = getAge() + 1;

}

return {

setAge: setAge,

incSalary: incSalary,

incAge: incAge,

setSalary: setSalary

setName: setName

}

})();

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

Employee = (function(){

var name;

var age;

var salary;

function getAge(){

return age;

}

function getSalary(){

return Salary;

}

function getName(){

return Name;

}

return{

setAge: setAge(newAge){

age = newAge;

}

setSalary: setSalary(newSalar){

age = newSalary;

}

setName: setName(newName){

age = newName;

}

incSalary:increaseSalary(percentage){

salary = getSalary() + getSalary\*percentage;

}

incAge:incrementAge(){

Age = getAge() + 1;

}

}

})();

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

Employee = (function(){

var name;

var age;

var salary;

function getAge(){

return age;

}

function getSalary(){

return Salary;

}

function getName(){

return Name;

}

var myObject = {};

myObject.setAge: setAge(newAge){

age = newAge;

}

myObject.setSalary: setSalary(newSalar){

age = newSalary;

}

myObject.setName: setName(newName){

age = newName;

}

myObject.incSalary:increaseSalary(percentage){

salary = getSalary() + getSalary\*percentage;

}

myObject.incAge:incrementAge(){

Age = getAge() + 1;

}

return myObject;

})();

1. 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;

getAddress:function getAddress(){

return address;

}

setAge: setAddress(newAddress){

address = newAadress;

}

}