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.

Answer:

Global Scope: is the root scope that is accessible and visible in all other scopes.

Local scope: is a function scope, any variable defined there is only accessible in it. It may be one of the following:

* Block scope: its accessibility between two curly brackets (use let and const)
* Function scope: accessible inside function (var)

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? 2

Answer:

1. No
2. Yes
3. No
4. Yes
5. Yes

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 add = (function() {

var counter = 0;

let print = function() {

console.log(counter)

}

return {

increase: function() {

counter++;

print(counter)

},

reset : function() {

counter = 0;

print(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?

Answer:

free variable is "counter"

Free variable: A variable referred to by a function that is not one of its parameters or local variables.

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 makeAdder = function (increment) {

var counter = 0;

return function () {

return counter += increment;

}

};

var add5 =makeAdder(5);

add5();

add5();

add5();

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? 3

Answer:  
By using module pattern

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 empModule = (function() {

var name

var age

var salary

var getAge = function getAge() {

return age;

}

var getName = function getAge() {

return name;

}

var getSalary = function getAge() {

return salary;

}

var setAge = function (age) {

this.age = age;

}

var setName = function (name) {

this.name = name;

}

var setSalary = function (salary) {

this.salary = salary;

}

var increaseSalary = function (percent) {

setSalary(getSalary \* (1 + percent / 100))

}

var incrementAge = function (years) {

setAge(getAge + years)

}

var print = function () {

var n = getName

console.log("name" + n + "age" + getAge + "salary" + getSalary)

}

return {

setAge: setAge,

setName: setName,

setSalary: setName,

increaseSalary: increaseSalary,

incrementAge: incrementAge,

print: print

}

})()

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

Answer:   
var module\_anonymous = (function() {

var name

var age

var salary

var getAge = function() {

return age;

}

var getName = function () {

return name;

}

var getSalary = function () {

return salary;

}

return {

setAge: function (age) {

this.age = age;

},

setName: function (name) {

this.name = name;

},

setSalary: function (salary) {

this.salary = salary;

},

increaseSalary: function (percent) {

setSalary(getSalary \* (1 + percent / 100))

},

incrementAge: function (years) {

setAge(getAge + years)

},

print: function print() {

var n = getName

console.log("name" + n + "age" + getAge + "salary" + getSalary)

}

};

})();

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

Answer:  
var stacked = (function() {

var name

var age

var salary

var getAge = function() {

return age;

}

var getName = function() {

return name;

}

var getSalary = function() {

return salary;

}

var emp = {

setAge: function(aage) {

age = aage;

},

setName: function(naame) {

name = naame;

},

setSalary: function(saalary) {

salary = saalary;

},

increaseSalary: function(percent) {

this.setSalary(getSalary \* (1 + percent / 100))

},

incrementAge: function(years) {

this.setAge(getAge + years)

},

print: function() {

var n = getName

console.log("name" + name + "age" + age + "salary" + salary)

}

}

return emp;

})();

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:

empModule.address;

empModule.setAddress = function(newAddress) {

return address = newAddress;

};

empModule.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:   
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)); 4

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