1. undefined
   1. 8
   2. 8
   3. 9
   4. 10
   5. 1
2. Variables that are outside the function are considered to be in the global scope, and can be accessed inside the function and outside the function, whereas local scope are variables that are declared inside a function or a block, and can only be accessed inside that function or block.

a)No

b)Yes

c)No

d)Yes

e)Yes

1. a. 81

b. 25

1. 10
2. //revealing module pattern
3. var count = (function() {
4. var counter = 0;
5. const getCounter = function() {
6. return counter;
7. }
8. const add = function() {
9. return counter += 1;
10. }
11. const reset = function() {
12. counter = 0;
13. }
14. return {
15. getCounter,
16. add,
17. reset
18. }
19. })();
20. Counter inside the function is a free variable. A free variable is a variable which is defined outside the block of execution. Instances created will have access to that free variable by reference even after execution. (closure).

make\_adder : function(inc){

return counter+inc;

}

1. Wrap the code in “module pattern”.

var student=(function(){

var name;

var age;

var salary;

function getAge(){

return this.age;

}

function getSalary(){

return this.salary;

}

function getName(){

return this.name;

}

function publicSetAge(newAge){

this.age=newAge;

}

function publicSetSalary(newSalary){

this.salary=newSalary;

}

function publicSetName(newName){

this.name=newName;

}

function publicIncreaseSalary(percentage){

this.salary=getSalary()+getSalary()\*percentage;

}

function publicIncrementAge(){

this.age+=1;

}

return{

setAge: publicSetAge,

setSalary:publicSetSalary,

setName:publicSetName,

increaseSalary:publicIncreaseSalary,

increamentAge:publicIncrementAge

};

})();

1. Student.prototype.address=”default”

Student.prototype.setAddress(newAddress){

This.address=newAddress;

}

Student.prototype.getAddress(){

Return this.address;

}