**Q1. Create a hierarchy of person, employee and developers.**

function person(){

this.name = 'Ananay';

this.age= 10;

}

function employees(){

this.id=1; }

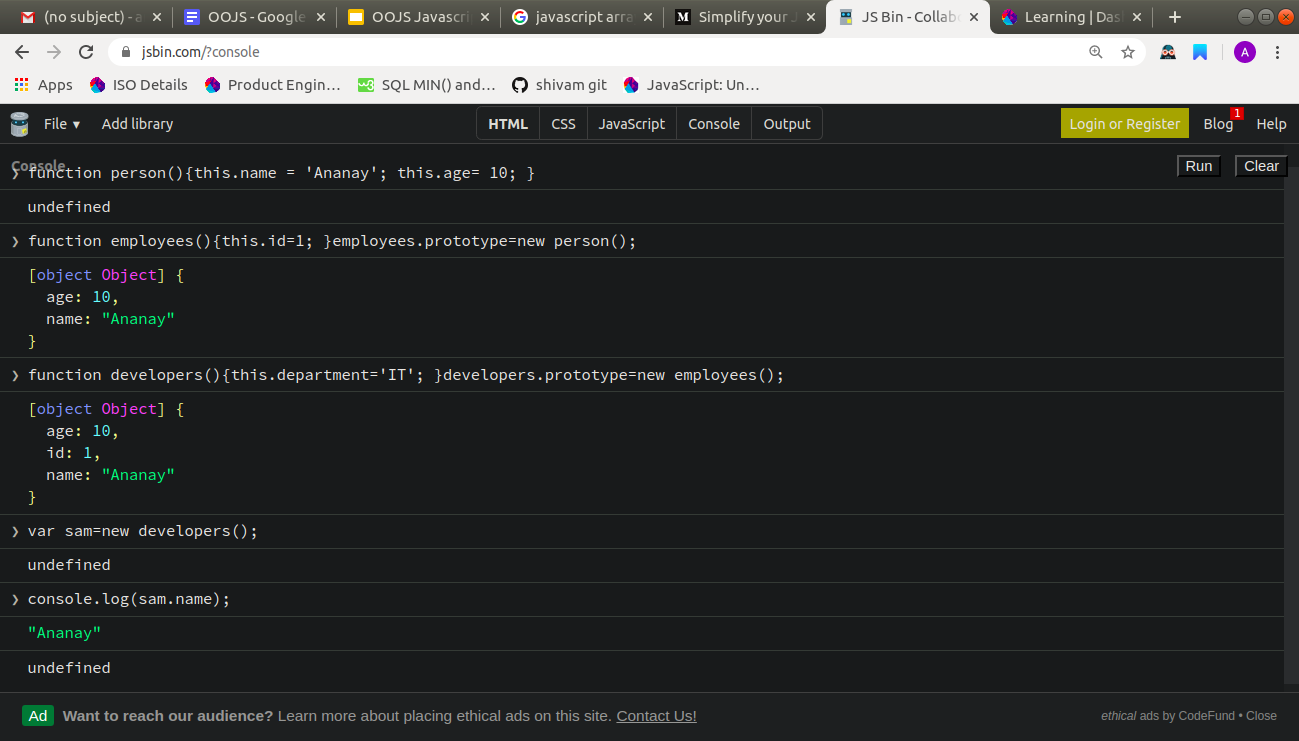
employees.prototype=new person();

function developers(){

this.department='IT'; }

developers.prototype=new employees();

Var sam= new developers();  
console.log(sam.name);



**Q2. Given an array, say [1,2,3,4,5]. Print each element of an array after 3 secs.**

var arr=[1,2,3,4,5];

for(var i=0;i<5;i++){

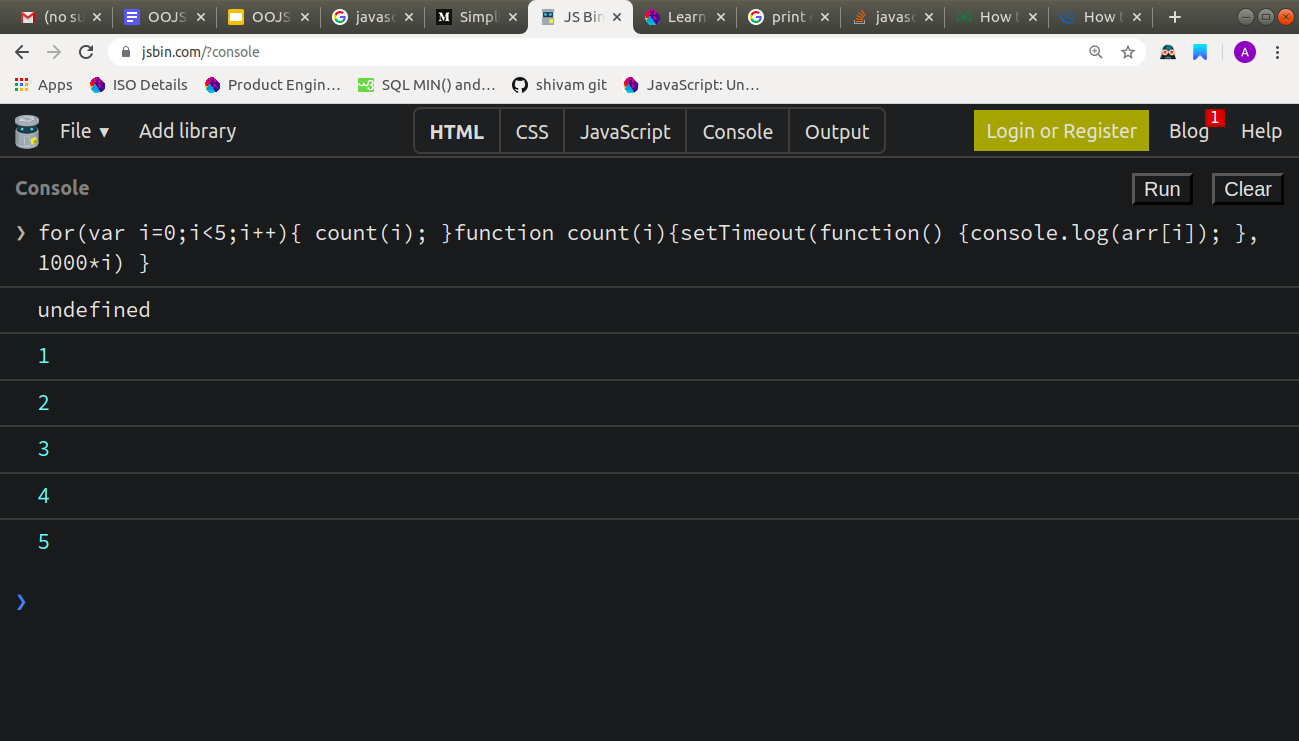
count(i);

}

function count(i){ setTimeout(function()

{console.log(arr[i]); }, 3000\*i)

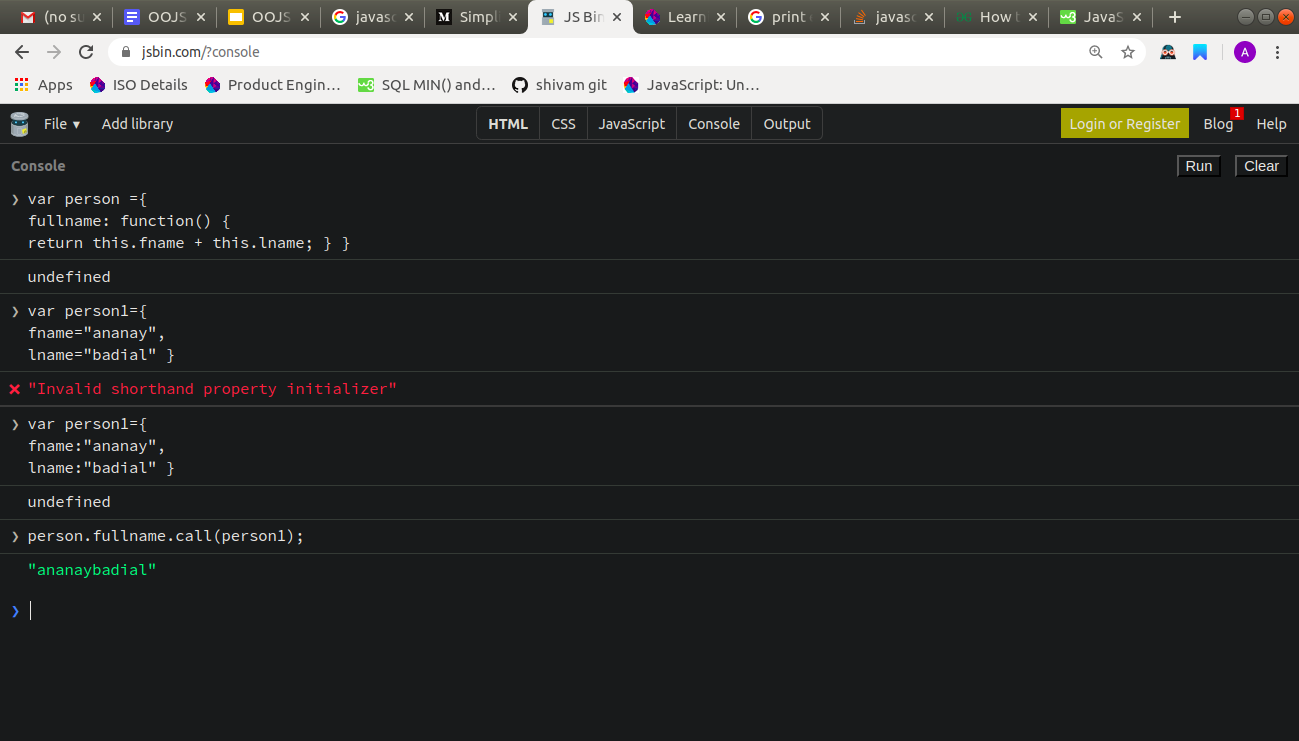
}



**Q3. Explain the difference between Bind and Call (example).**

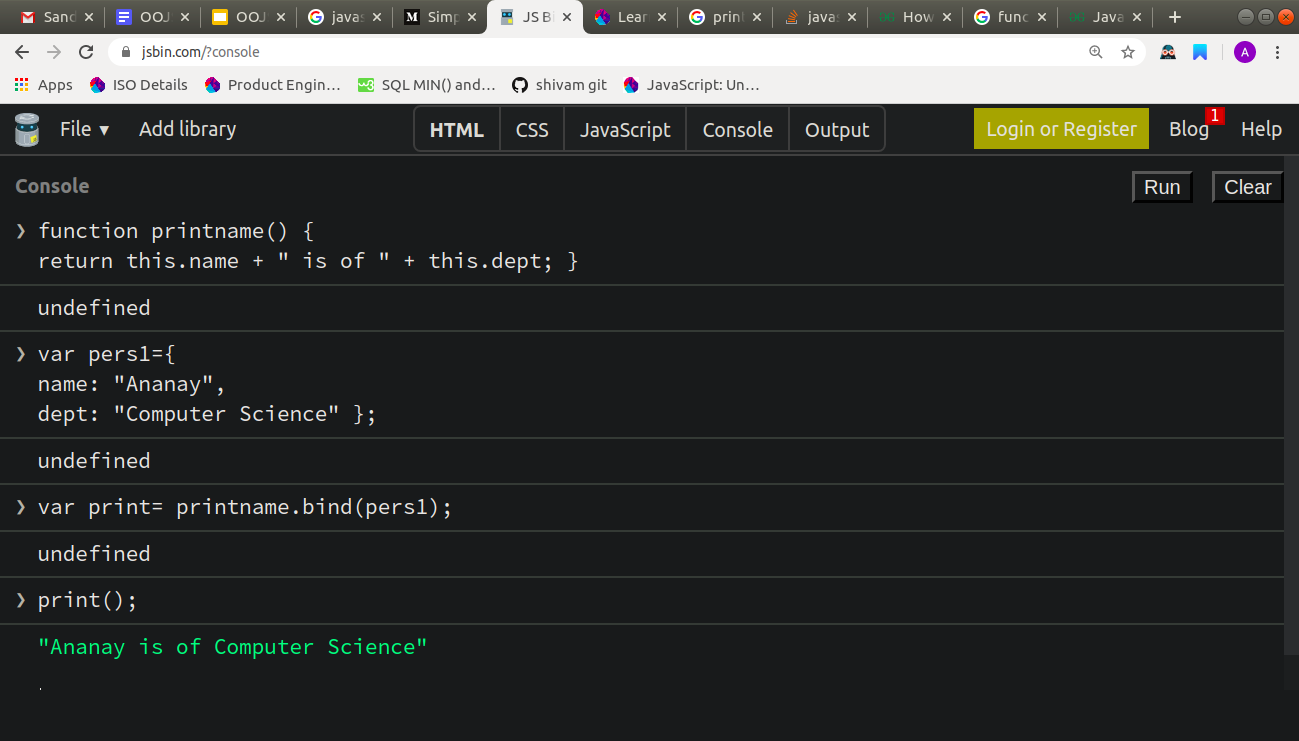
Call attaches this into function and executes the function immediately

For example : -



Bind attaches this into function and it needs to be invoked separately.

For example :-



**Q4. Explain 3 properties of argument object.**

A.Property of argument object :

1. argument.callee

Callee is a property of the arguments object. It can be used to refer to the currently executing function inside the function body of that function. This is useful when the name of the function is unknown, such as within a function expression with no name (also called "anonymous functions").

2 .argument.length

The arguments.length property provides the number of arguments actually passed to a function. This can be more or less than the defined parameter's count.

3. arguments[@@iterator]() The initial value of the @@iterator property is the same function object as the initial value of the Array.prototype.values property.

Q5. Create a function which returns number of invocations and number of instances of a function.

invocationAndInstanceCounter.instances = 0;

invocationAndInstanceCounter.invocations = 0;

function invocationAndInstanceCounter() {

if (this instanceof invocationAndInstanceCounter) {

invocationAndInstanceCounter.instances += 1;

} else {

invocationAndInstanceCounter.invocations += 1;

}

console.log(

"invocations: " + invocationAndInstanceCounter.invocations + " instances: " + invocationAndInstanceCounter.instances

);

}

invocationAndInstanceCounter();

new invocationAndInstanceCounter();

invocationAndInstanceCounter();

new invocationAndInstanceCounter();

invocationAndInstanceCounter();

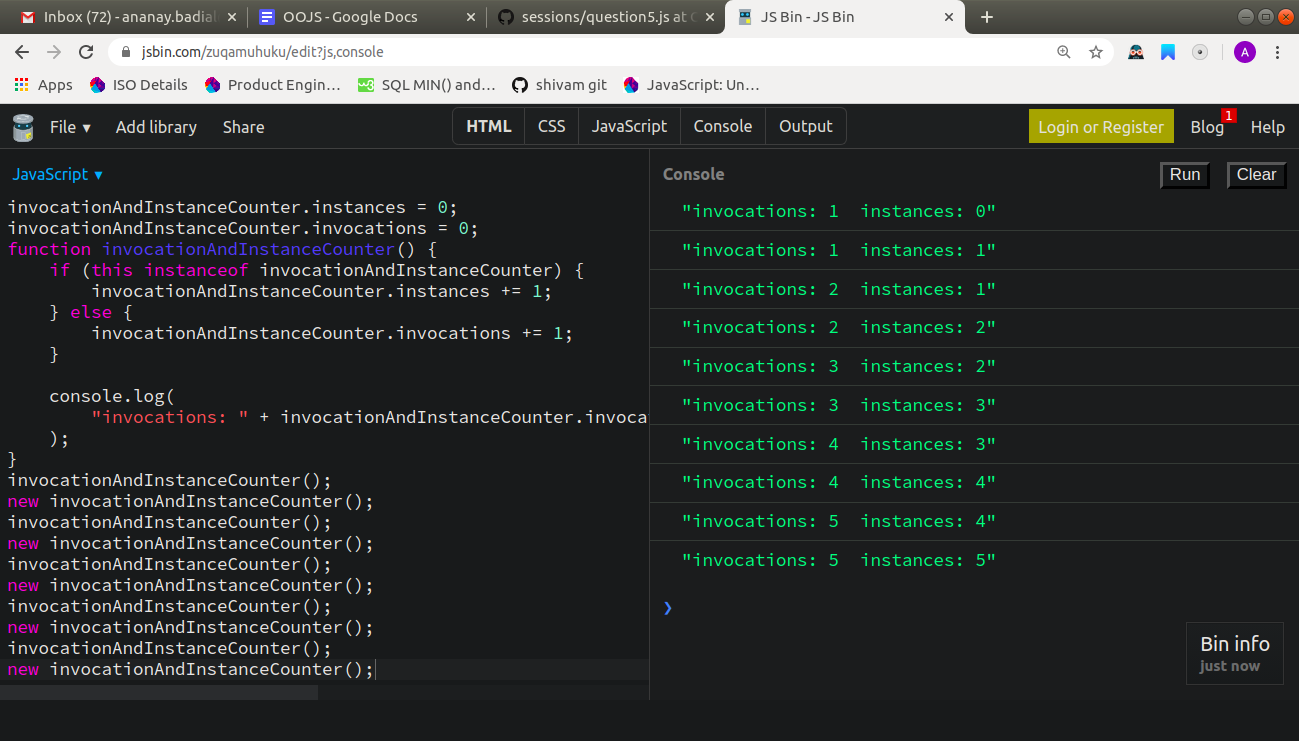
new invocationAndInstanceCounter();

invocationAndInstanceCounter();

new invocationAndInstanceCounter();

invocationAndInstanceCounter();

new invocationAndInstanceCounter();



6. Q6. Create a counter using closures.

var uniqueint = (function() {

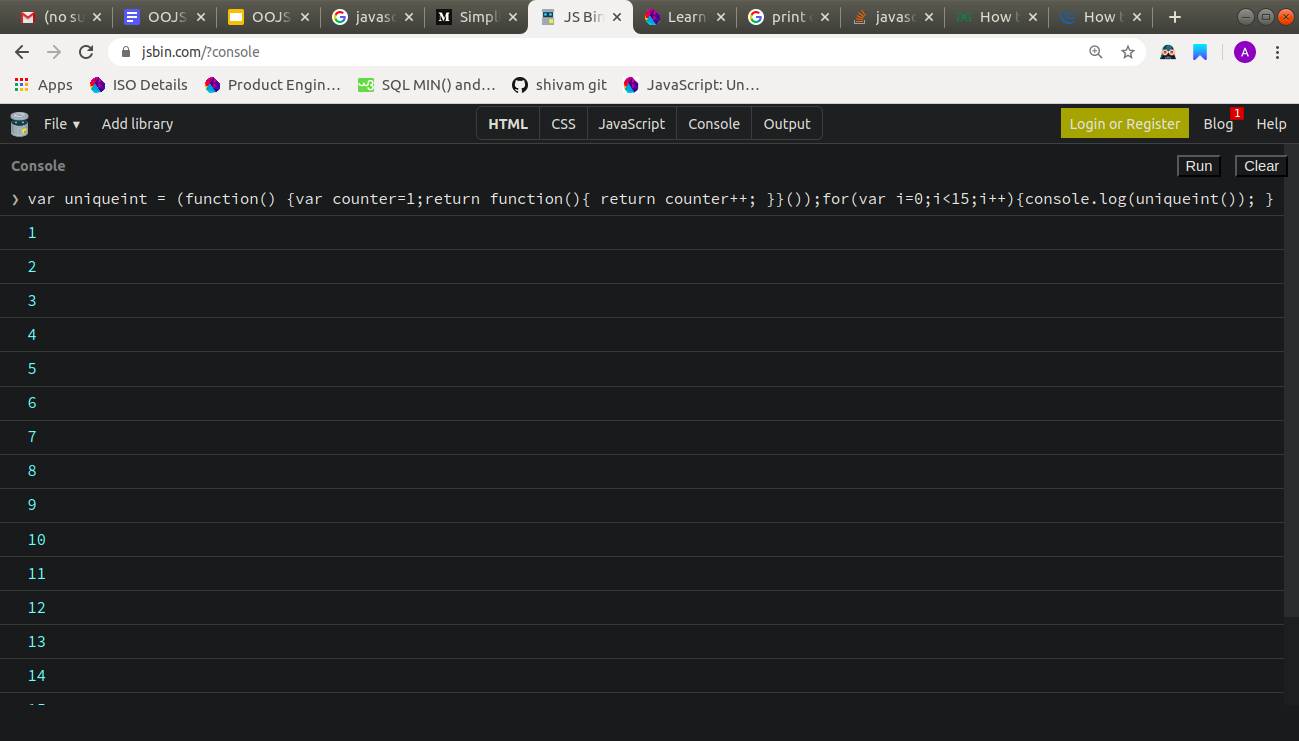
var counter=1;

return function(){ return counter++; }

}());

for(var i=0;i<15;i++){

console.log(uniqueint()); }



**Q7. Explain 5 array methods with example.**

A.

1. forEach()

The forEach() method calls a function once for each element in an array, in order.

var sum=0;

var num=[20,30,40,50];

num.forEach(Sumfunc);

function Sumfunc(item){

sum+=item; }

console.log(“Sum = “ + sum); //Sum = 140

2. map()  
 Creates a new array with the result of calling a function for each array element.  
  
 var numbers=[2,3,4,5];

var newnum=numbers.map(prodfun);

function prodfun(item){

return item\*10; }

console.log(newnum); // [20,30,40,50]

3. reduce()  
 The reduce() method reduces the array to a single value.

var numbers=[48,12,16,10];

var newnum=numbers.reduce(myFunc);

function myFunc(total,item){

return total-item; }

console.log(newnum); //10

4. valueOf()

The valueOf() method returns the array.  
  
var fruits = ["Banana", "Orange", "Apple", "Mango"];

var v = fruits.valueOf(); // ["Banana", "Orange", "Apple", "Mango"]

5. Push()  
  
Pushes the element into the array at the end position and returns the length of array  
  
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.push("Kiwi"); // 5