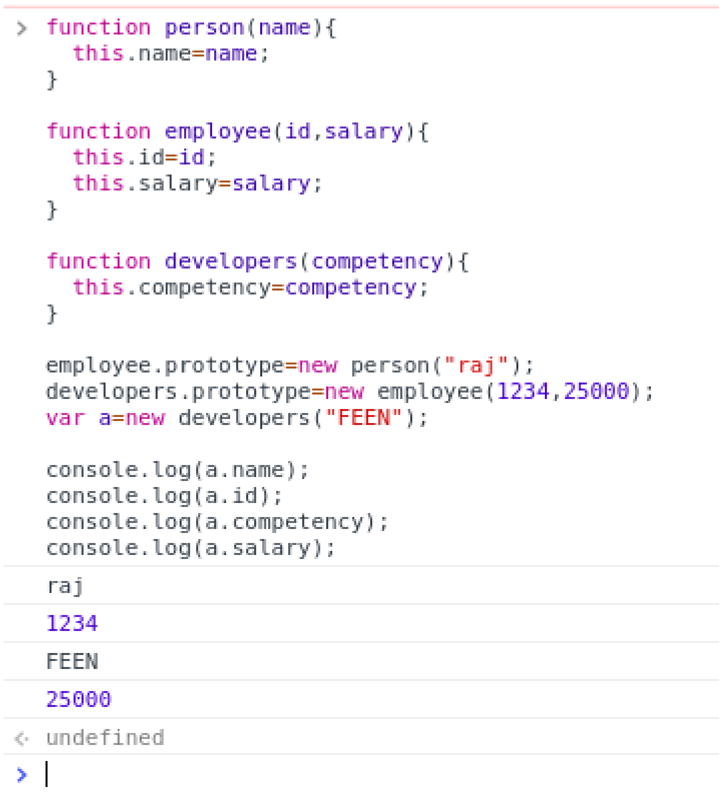
Que 1:- **Create a hierarchy of person, employee and developers.**

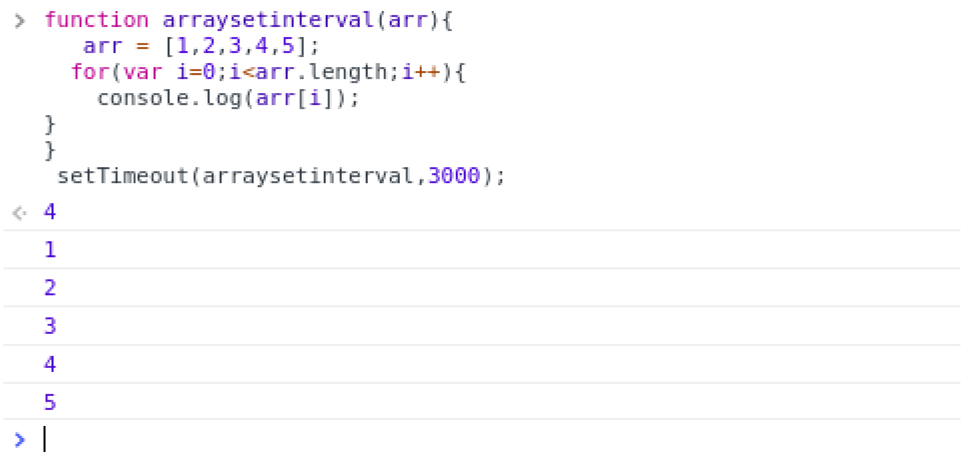
Ans:- Output :-



```
> function person(name){
    this.name=name;
  }

  function employee(id,salary){
    this.id=id;
    this.salary=salary;
  }

  function developers(competency){
    this.competency=competency;
  }

  employee.prototype=new person("raj");
  developers.prototype=new employee(1234,25000);
  var a=new developers("FEEN");

  console.log(a.name);
  console.log(a.id);
  console.log(a.competency);
  console.log(a.salary);
  raj
  1234
  FEEN
  25000
<· undefined
> |
```

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

Ans :-



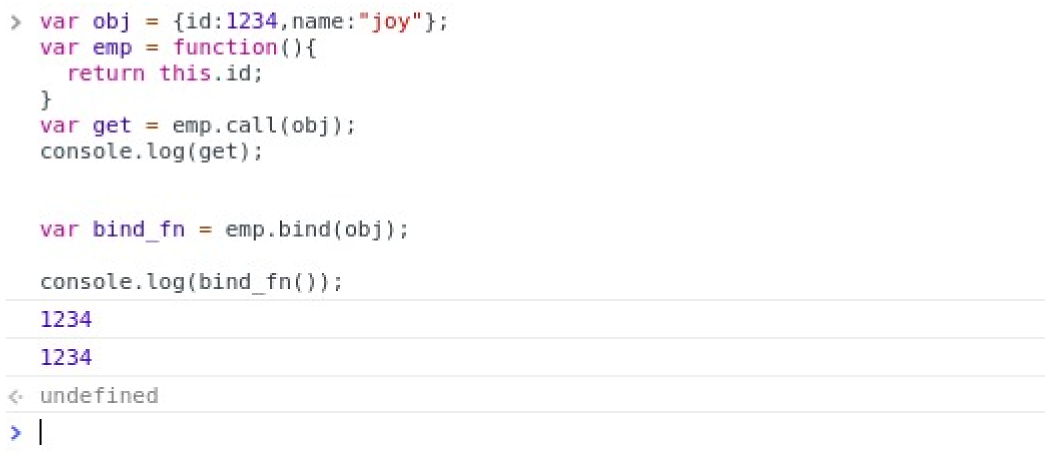
```
> function arraysetinterval(arr){
     arr = [1,2,3,4,5];
    for(var i=0;i<arr.length;i++){
      console.log(arr[i]);
  }
  }
   setTimeout(arraysetinterval,3000);
<· 4
  1
  2
  3
  4
  5
> |
```

Que : 3 :- **Explain difference between Bind and Call (example)**

Ans:-

Call :- Call directly return the value

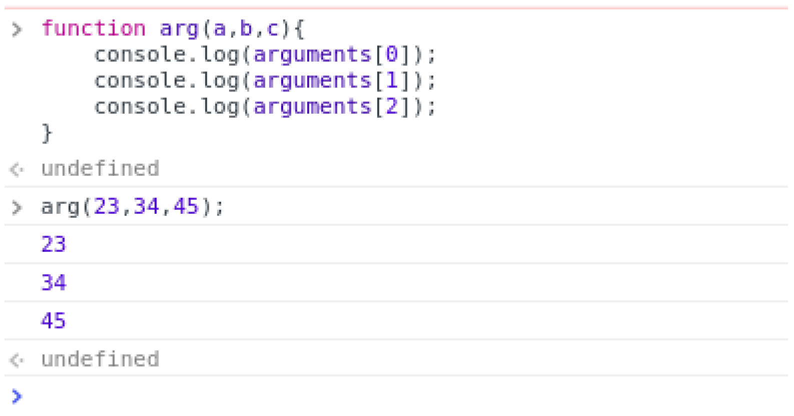
bind :- bind return the function which is refering the value.so we access the value after calling the bind function.



```
> var obj = {id:1234,name:"joy"};
  var emp = function(){
    return this.id;
  }
  var get = emp.call(obj);
  console.log(get);


  var bind_fn = emp.bind(obj);

  console.log(bind_fn());
  1234
  1234
< undefined
> |
```

Que : 4 :- Explain 3 properties of argument object.

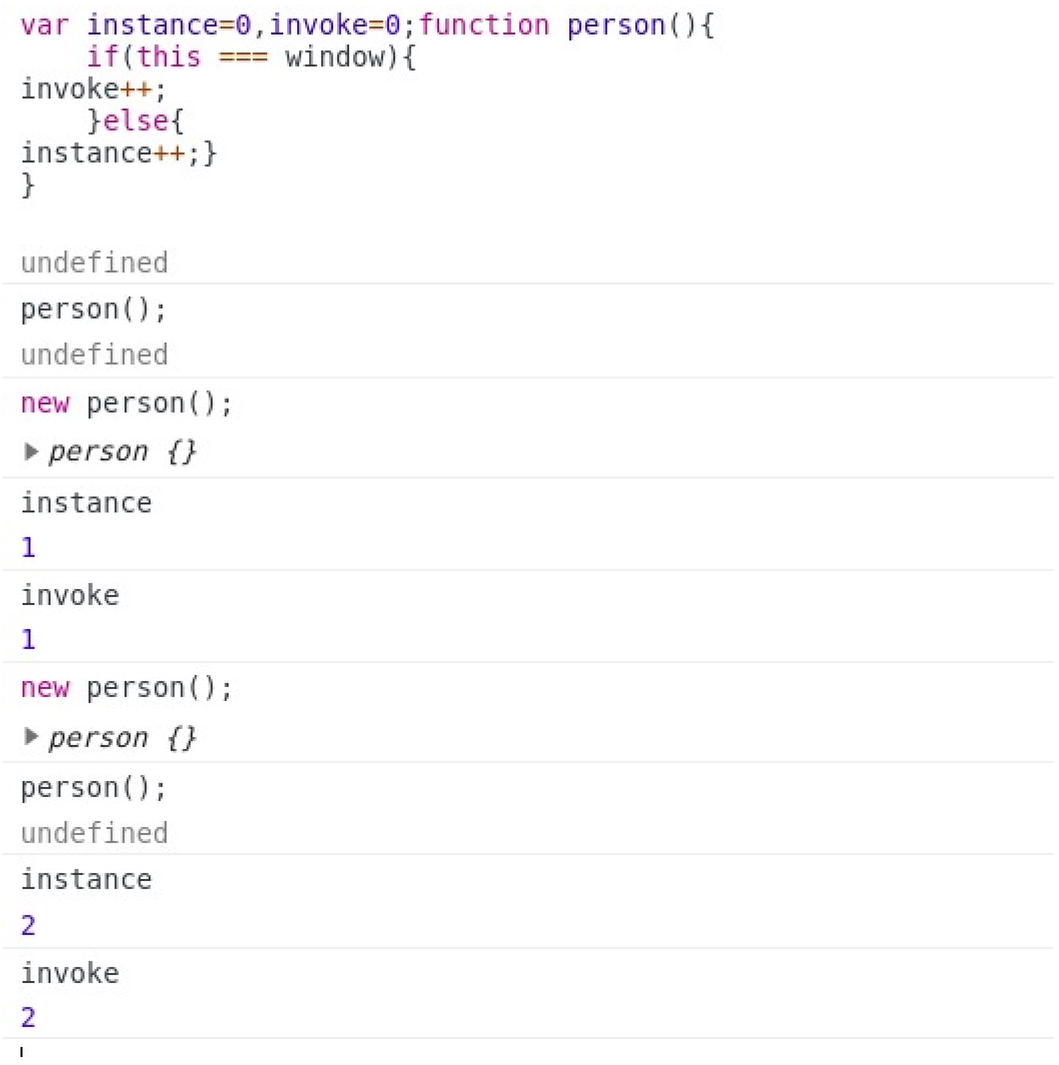
Ans:-



```
> function arg(a,b,c){
      console.log(arguments[0]);
      console.log(arguments[1]);
      console.log(arguments[2]);
  }
< undefined
> arg(23,34,45);
  23
  34
  45
< undefined
>
```

Que 5 :- Create a function which returns number of invocations and number of instances of a function.

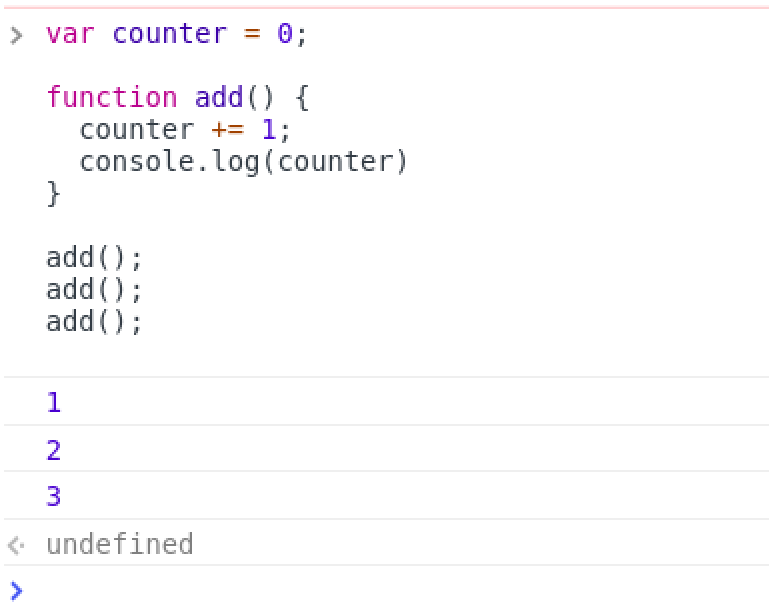
Ans:-



```
var instance=0,invoke=0;function person(){
    if(this === window){
invoke++;
    }else{
instance++;}
}

undefined
person();
undefined
new person();
▶ person {}
instance
1
invoke
1
new person();
▶ person {}
person();
undefined
instance
2
invoke
2
```

Que 6:- Create a counter using closures.

Ans:-



```
> var counter = 0;

  function add() {
    counter += 1;
    console.log(counter)
  }

  add();
  add();
  add();

  1
  2
  3
<· undefined
>
```