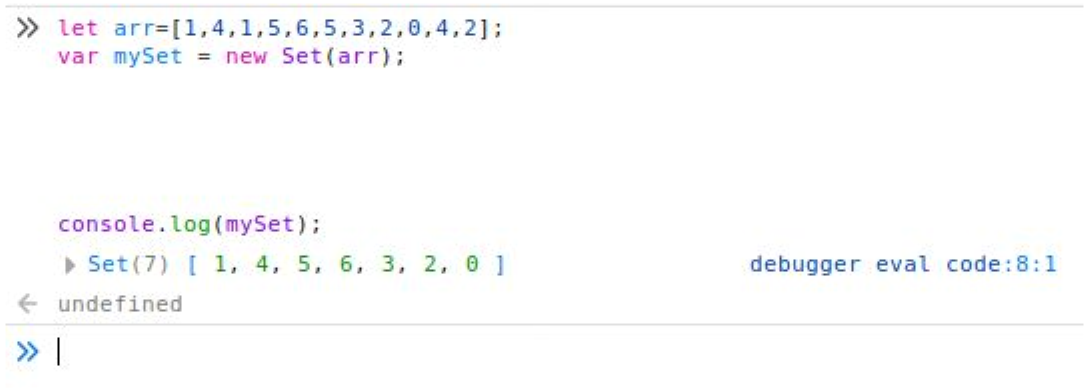
1. **Filter unique array members using Set.**

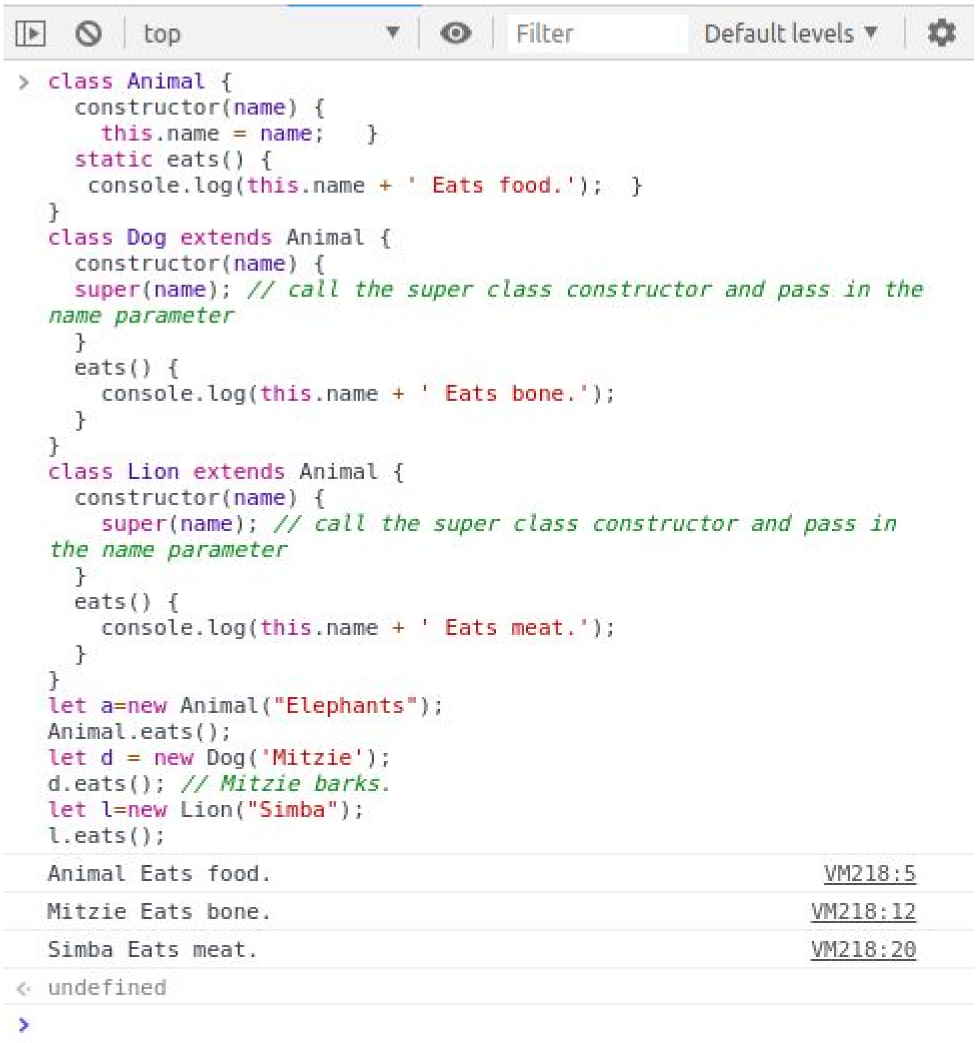


```
>> let arr=[1,4,1,5,6,5,3,2,0,4,2];
   var mySet = new Set(arr);




   console.log(mySet);
   ▸ Set(7) [ 1, 4, 5, 6, 3, 2, 0 ]                debugger eval code:8:1
← undefined
>> |
```

2. **Write a program to implement inheritance upto 3 classes.The Class must public variables and static functions.**

   Ans.

Inheritance in most class-based object-oriented languages is a mechanism in which one object acquires all the properties and behaviors of another object. JavaScript is not a class-based language although class keyword is introduced in ES2015, it is just syntactical layer. JavaScript still works on prototype chain.

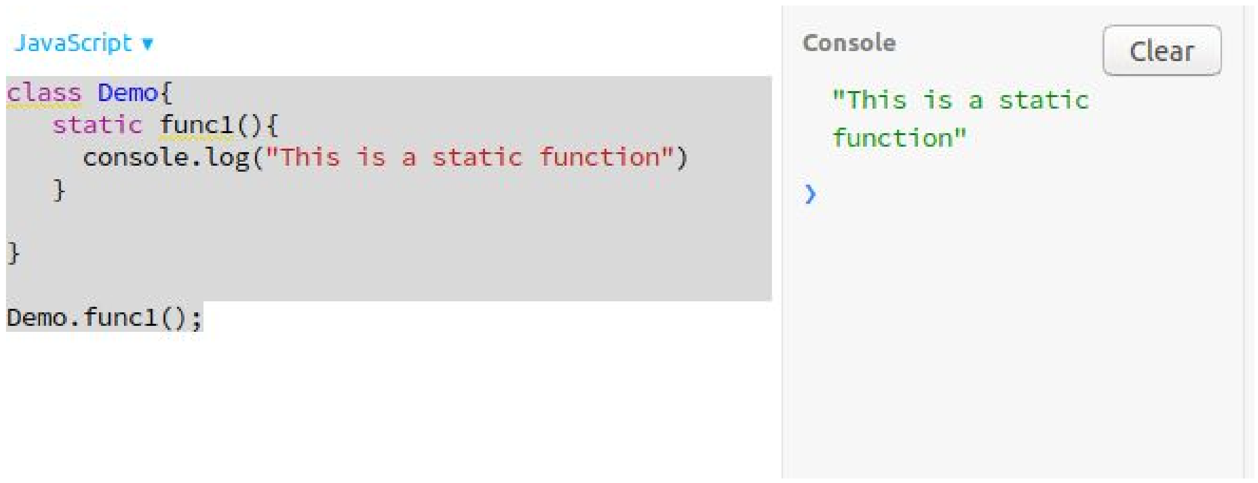


```
> class Animal {
    constructor(name) {
      this.name = name;    }
    static eats() {
     console.log(this.name + ' Eats food.');  }
  }
  class Dog extends Animal {
    constructor(name) {
    super(name); // call the super class constructor and pass in the
  name parameter
    }
    eats() {
      console.log(this.name + ' Eats bone.');
    }
  }
  class Lion extends Animal {
    constructor(name) {
      super(name); // call the super class constructor and pass in
  the name parameter
    }
    eats() {
      console.log(this.name + ' Eats meat.');
    }
  }
  let a=new Animal("Elephants");
  Animal.eats();
  let d = new Dog('Mitzie');
  d.eats(); // Mitzie barks.
  let l=new Lion("Simba");
  l.eats();
  Animal Eats food.                                        VM218:5
  Mitzie Eats bone.                                        VM218:12
  Simba Eats meat.                                         VM218:20
< undefined
>
```

3. **Write a program to implement a class having static functions**

```
class Demo{
  static func1(){
console.log("This is a static function")
  }
}
```

Demo.func1();



```
class Demo{
   static func1(){
     console.log("This is a static function")
   }

}

Demo.func1();
```

```
"This is a static
function"
›
```

4. **Import a module containing the constants and method for calculating area of circle, rectangle, cylinder.**

   Ans.

   module1.js

```
export function AreaOfCircle(r){

   return 3.14*(r*r);

 }

 export function AreaOfRect(l,b){

   return l*b;

 }

 export function AreaOfCylinder(h,r){

   return (2*3.14*r*h+2*3.14*r*r);
 }
```

Exec.js

```
import {AreaOfRect,AreaOfCircle,AreaOfCylinder} from './module1';

document.getElementById("alpha").innerHTML=AreaOfCircle(5);
```

5. **Import a module for filtering unique elements in an array.**

module2.js

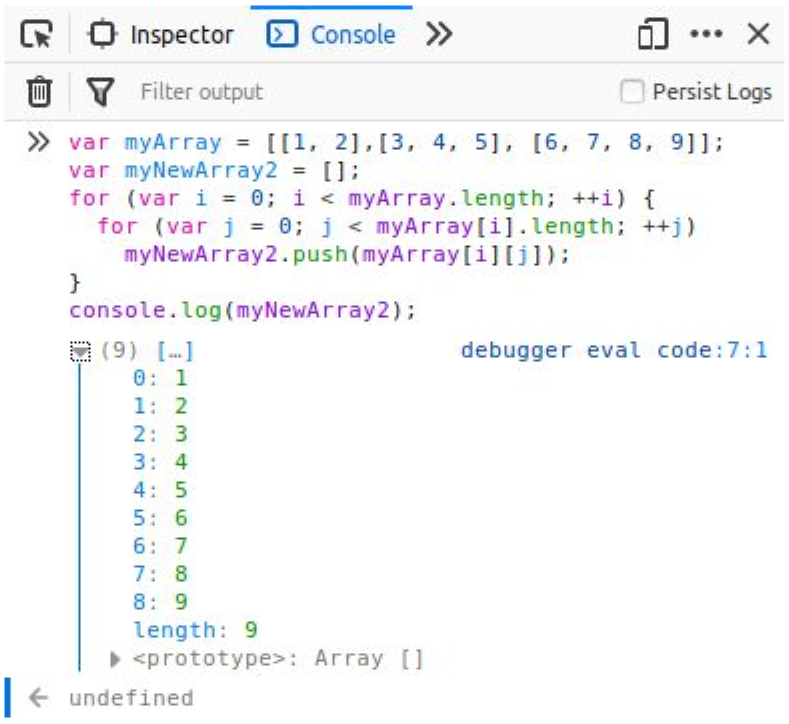
```
export function onlyUnique(value, index,self) {

   return self.indexOf(value) === index;

}

console.log(unique)
```

Exec2.js

```
import {filterModule} from './module2';

// usage example:

var a = ['a', 1, 'a', 2, '1'];

var unique = a.filter( onlyUnique );
```

6. **Write a program to flatten a nested array to single level using arrow functions.**

```
var myArray = [[1, 2],[3, 4, 5], [6, 7, 8, 9]];

var myNewArray2 = [];

for (var i = 0; i < myArray.length; ++i) {

  for (var j = 0; j < myArray[i].length; ++j)

    myNewArray2.push(myArray[i][j]);

}

console.log(myNewArray2);
```

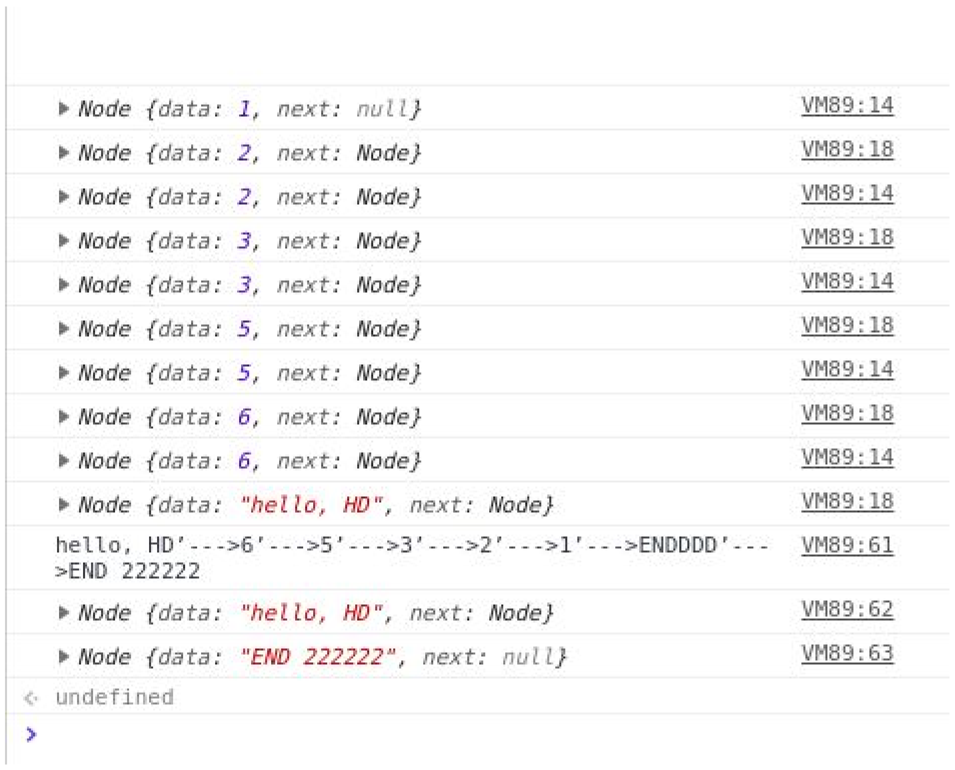


```
>> var myArray = [[1, 2],[3, 4, 5], [6, 7, 8, 9]];
   var myNewArray2 = [];
   for (var i = 0; i < myArray.length; ++i) {
     for (var j = 0; j < myArray[i].length; ++j)
       myNewArray2.push(myArray[i][j]);
   }
   console.log(myNewArray2);
   (9) […]                     debugger eval code:7:1
       0: 1
       1: 2
       2: 3
       3: 4
       4: 5
       5: 6
       6: 7
       7: 8
       8: 9
       length: 9
     ▶ <prototype>: Array []
<- undefined
```

7. **Implement a linked list in es6 and implement addFirst() addLast(), length(), getFirst(), getLast().**

Ans.

```
class Node{
 constructor(value){
   this.data = value;
   this.next = null;
 }
}
class LinkedList{
 constructor(value){
   this.head = new Node(value);
   this.tail = this.head;
 }

 addFirst(value){
  console.log(this.head);
   let new_node = new Node(value);
   new_node.next = this.head;
   this.head = new_node;
   console.log(this.head);
 }
 addLast(value){
   let new_node = new Node(value);
   this.tail.next = new_node;
   this.tail = new_node;
 }
 length(){
   let temp_head = this.head;
   let length = 0;
   while(temp_head.next != null){
           length++;
           temp_head = temp_head.next;
   }
  return length;
 }
 getFirst(){
  return this.head;
 }
 getLast(){
  return this.tail;
 }
 toString = () => {
  let temp_head = this.head;
  let str = '';
  while(temp_head != null){
          str += `${temp_head.data}`;
          if(temp_head.next!=null){
          str+=`’---> `
          }
          temp_head = temp_head.next;
  }
  return str;
 }
}
let my_linkedList = new LinkedList(1);
my_linkedList.addFirst(2);
my_linkedList.addFirst(3);
my_linkedList.addFirst(5);
my_linkedList.addFirst(6);
my_linkedList.addFirst('hello, HD');
my_linkedList.addLast("ENDDDD");
my_linkedList.addLast("END 222222");
console.log(my_linkedList + '');
console.log(my_linkedList.getFirst());
console.log(my_linkedList.getLast());
```



```
▶ Node {data: 1, next: null}                           VM89:14
▶ Node {data: 2, next: Node}                           VM89:18
▶ Node {data: 2, next: Node}                           VM89:14
▶ Node {data: 3, next: Node}                           VM89:18
▶ Node {data: 3, next: Node}                           VM89:14
▶ Node {data: 5, next: Node}                           VM89:18
▶ Node {data: 5, next: Node}                           VM89:14
▶ Node {data: 6, next: Node}                           VM89:18
▶ Node {data: 6, next: Node}                           VM89:14
▶ Node {data: "hello, HD", next: Node}                 VM89:18
hello, HD'--->6'--->5'--->3'--->2'--->1'--->ENDDDD'---  VM89:61
>END 222222
▶ Node {data: "hello, HD", next: Node}                 VM89:62
▶ Node {data: "END 222222", next: null}                VM89:63
<· undefined
>
```

8. **Implement Map and Set using Es6?**

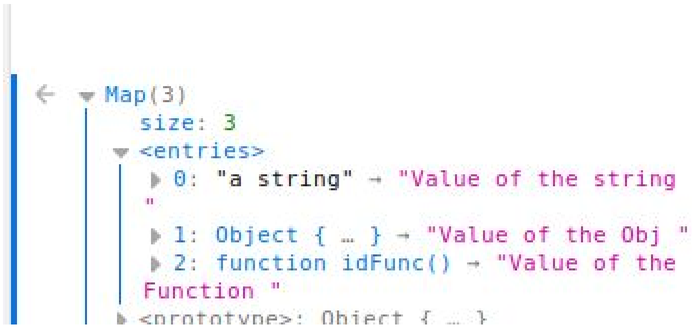
Ans.

The Map object holds key-value pairs and remembers the original insertion order of the keys. Any value (both objects and primitive values) may be used as either a key or a value.

Code:

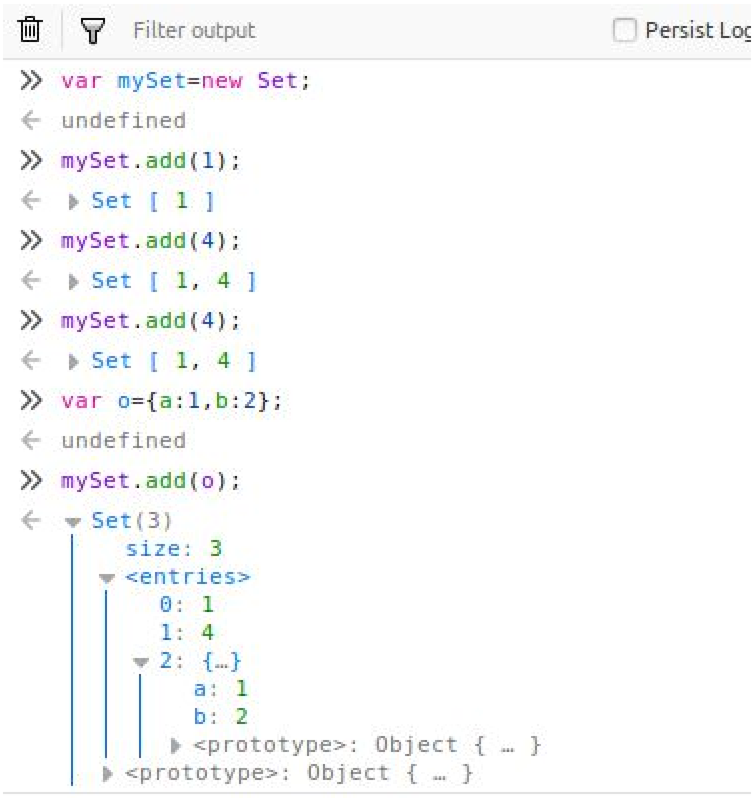
```
 var myMap =new Map();
var idString="a string",
idObj={1:'Abbie'},
idFunc=function(){};
```

```
 myMap.set(idString,"Value of the string ");
myMap.set(idObj,"Value of the Obj ");
 myMap.set(idFunc,"Value of the Function ");
```



```
← ▼ Map(3)
      size: 3
    ▼ <entries>
      ▶ 0: "a string" → "Value of the string
      "
      ▶ 1: Object { … } → "Value of the Obj "
      ▶ 2: function idFunc() → "Value of the
      Function "
    ▶ <prototype>: Object { … }
```

The Set object lets you store unique values of any type, whether primitive values or object references.
Syntax: new Set(iterable)



```
>> var mySet=new Set;
←  undefined
>> mySet.add(1);
←  ▶ Set [ 1 ]
>> mySet.add(4);
←  ▶ Set [ 1, 4 ]
>> mySet.add(4);
←  ▶ Set [ 1, 4 ]
>> var o={a:1,b:2};
←  undefined
>> mySet.add(o);
←  ▼ Set(3)
       size: 3
     ▼ <entries>
         0: 1
         1: 4
       ▼ 2: {…}
            a: 1
            b: 2
          ▶ <prototype>: Object { … }
     ▶ <prototype>: Object { … }
```

9. **Implementation of stack ?**

Ans.

```
class Stack {

      constructor()

    {

        this.items = [];

    }

  push(element)

  {

    this.items.push(element);

  }

  pop()

  {

    if (this.items.length == 0)

        return "Underflow";

    return this.items.pop();

  }

  peek()
{
      return this.items[this.items.length - 1];
}
isEmpty()
{
    return this.items.length == 0;
}
printStack()
{
    var str = "";
    for (var i = 0; i < this.items.length; i++)
        str += this.items[i] + " ";
    return str;
}


}
var stack = new Stack();
 console.log(stack.isEmpty());


console.log(stack.pop());
stack.push(10);
stack.push(20);
```

stack.push(30);

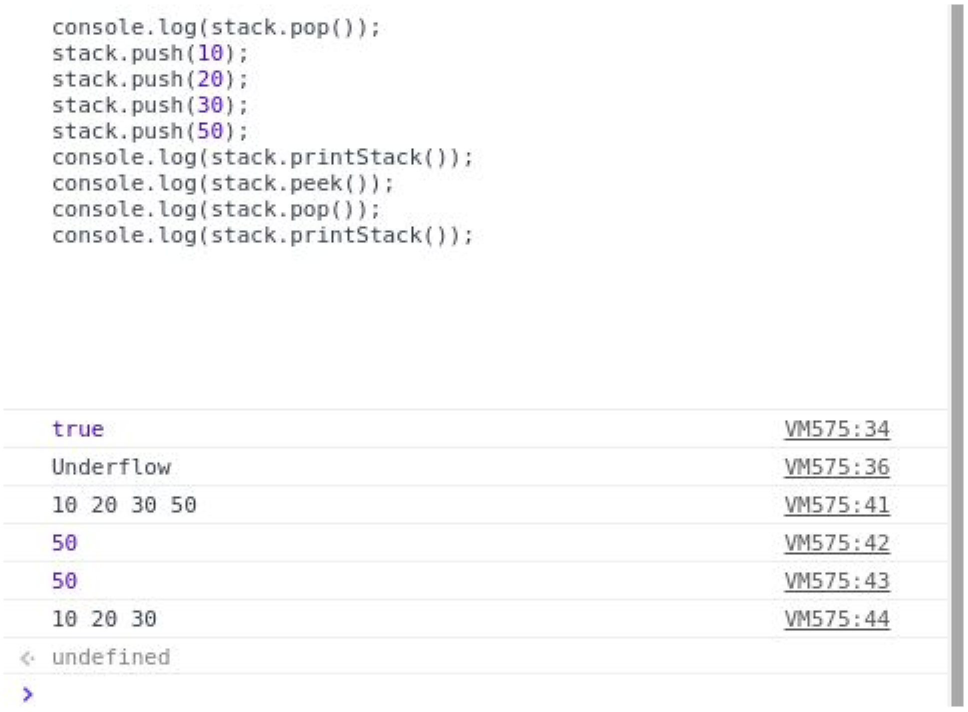
stack.push(50);

console.log(stack.printStack());

console.log(stack.peek());

console.log(stack.pop());

console.log(stack.printStack());



```
console.log(stack.pop());
stack.push(10);
stack.push(20);
stack.push(30);
stack.push(50);
console.log(stack.printStack());
console.log(stack.peek());
console.log(stack.pop());
console.log(stack.printStack());
```

```
true                                   VM575:34
Underflow                              VM575:36
10 20 30 50                            VM575:41
50                                     VM575:42
50                                     VM575:43
10 20 30                               VM575:44
< undefined
>
```