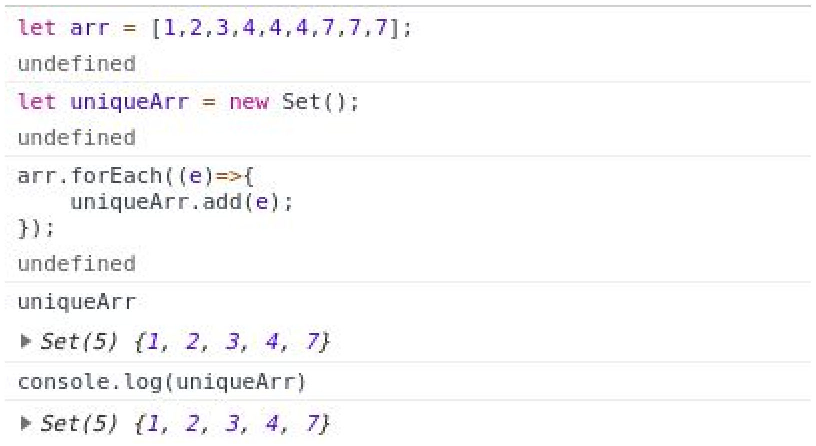
ES6 Javascript Assessment - 2

1. Filter unique array members using Set.

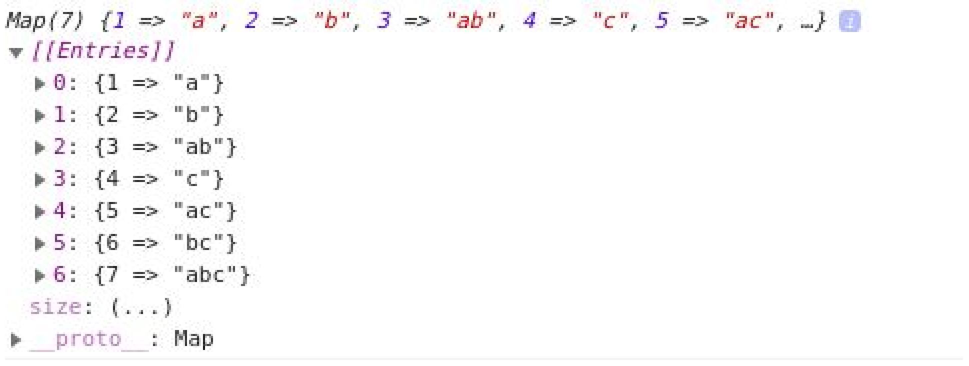
```
let arr = [1,2,3,4,4,4,7,7,7];

let uniqueArr = new Set();

arr.forEach((e)=>{

 uniqueArr.add(e);

});

console.log(uniqueArr)
```



```
let arr = [1,2,3,4,4,4,7,7,7];
undefined
let uniqueArr = new Set();
undefined
arr.forEach((e)=>{
    uniqueArr.add(e);
});
undefined
uniqueArr
▶ Set(5) {1, 2, 3, 4, 7}
console.log(uniqueArr)
▶ Set(5) {1, 2, 3, 4, 7}
```

2. Find the possible combinations of a string and store them in a MAP?

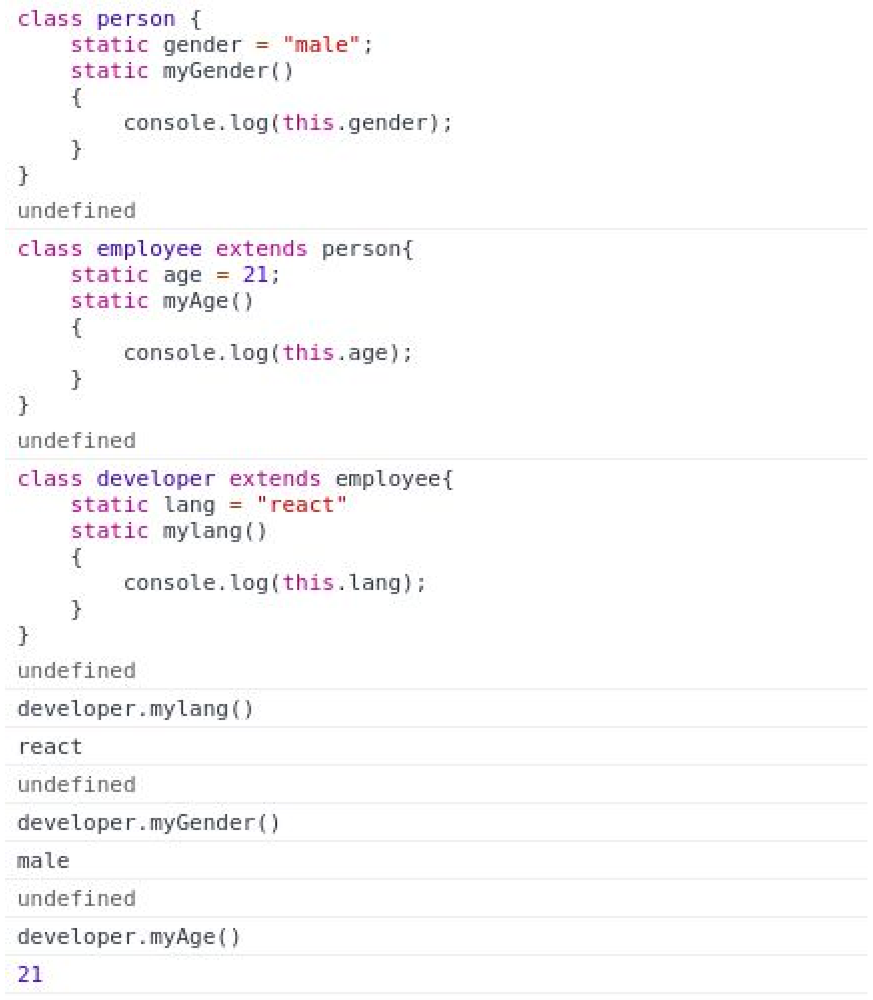
```
const str = 'abc';
const allCombinations = (str1) => {
   const arr = [];
   for (let x = 0, y=1; x < str1.length; x++,y++) {
      arr[x]=str1.substring(x, y);
   };
   const mp = new Map();
   let temp= "";
   let len = Math.pow(2, arr.length);
   for (let i = 0; i < len ; i++){
      temp= "";
      for (let j=0;j<arr.length;j++) {
         if ((i & Math.pow(2,j))){
            temp += arr[j];
         }
      };
      if (temp !== ""){
           mp.set(i,temp);
      }
   }
   return mp;
};
console.log(allCombinations(str));
```



```
Map(7) {1 => "a", 2 => "b", 3 => "ab", 4 => "c", 5 => "ac", …}
▼ [[Entries]]
  ▶ 0: {1 => "a"}
  ▶ 1: {2 => "b"}
  ▶ 2: {3 => "ab"}
  ▶ 3: {4 => "c"}
  ▶ 4: {5 => "ac"}
  ▶ 5: {6 => "bc"}
  ▶ 6: {7 => "abc"}
  size: (...)
▶ __proto__: Map
```

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

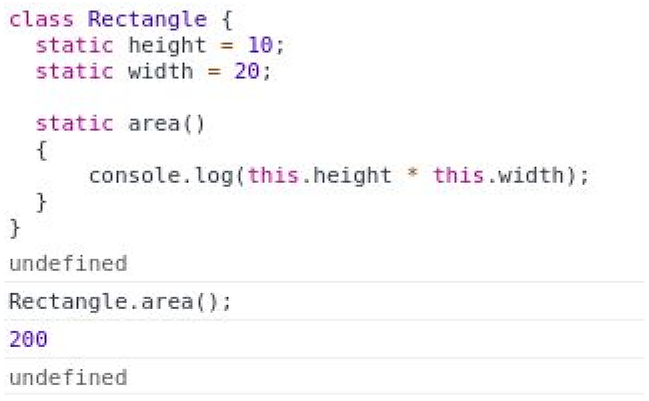
```
class person {

    static gender = "male";

    static myGender()

    {

        console.log(this.gender);

    }

}

class employee extends person{

    static age = 21;

    static myAge()

    {

        console.log(this.age);

    }

}

class developer extends employee{

    static lang = "react"

    static mylang()
    {
        console.log(this.lang);
    }
}


developer.mylang()

developer.myGender()

developer.myAge()
```



```
class person {
    static gender = "male";
    static myGender()
    {
        console.log(this.gender);
    }
}
undefined
class employee extends person{
    static age = 21;
    static myAge()
    {
        console.log(this.age);
    }
}
undefined
class developer extends employee{
    static lang = "react"
    static mylang()
    {
        console.log(this.lang);
    }
}
undefined
developer.mylang()
react
undefined
developer.myGender()
male
undefined
developer.myAge()
21
```

4. Write a program to implement a class having static functions.

```
class Rectangle {

  static height = 10;

  static width = 20;


  static area()

  {

      console.log(this.height * this.width);

  }

}
```



```
class Rectangle {
  static height = 10;
  static width = 20;

  static area()
  {
      console.log(this.height * this.width);
  }
}
undefined
Rectangle.area();
200
undefined
```

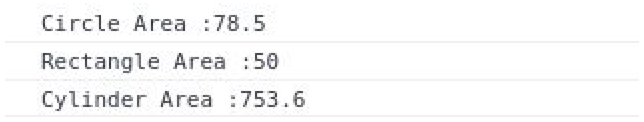
5. Import a module containing the constants and method for calculating area of circle, rectangle, cylinder.

exportAreas.js

```
const PI = 3.14;

export function circleArea(r)

{

    return PI * r * r;


}

export function rectArea(l,b)

{

    return l * b;

}

export function cylinArea(r,h)

{

    return (2 * PI * r * h) + (2 * PI * r * r);

}
```

App.js

```
import {circleArea,rectArea,cylinArea} from './exportAreas.js';

console.log("Circle Area :" + circleArea(5));

console.log("Rectangle Area :" + rectArea(5,10));

console.log("Cylinder Area :" + cylinArea(10,2));
```



```
Circle Area :78.5
Rectangle Area :50
Cylinder Area :753.6
```

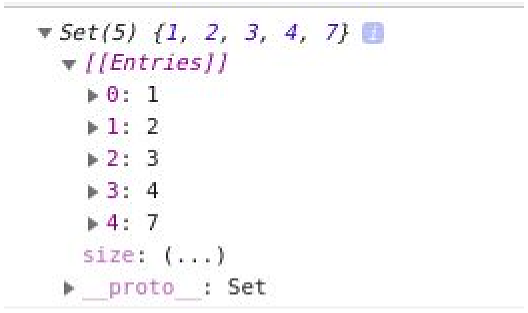
6. Import a module for filtering unique elements in an array.

uniqueArrEle.js

```
export function uniqArr(arr)

{

    let uniqueArr = new Set();

      arr.forEach((e)=>{

       uniqueArr.add(e);

    });

    console.log(uniqueArr)

}
```

App.js

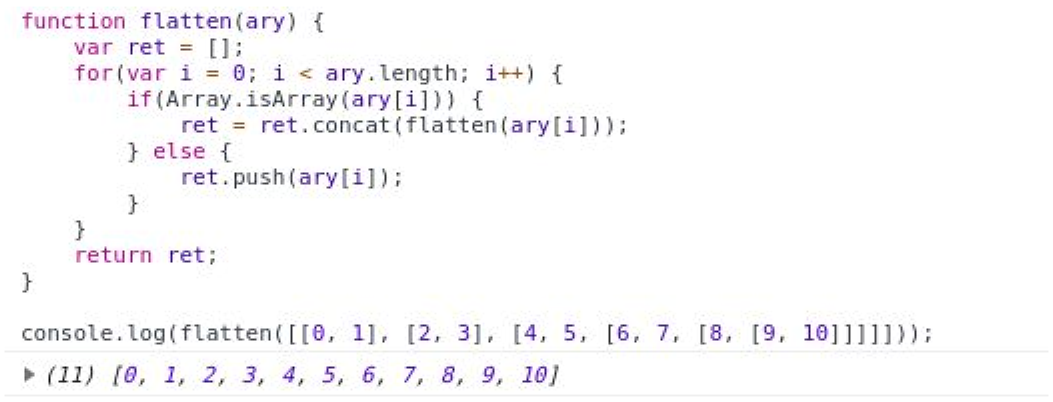
```
import {uniqArr} from './uniqueArrEle.js'

let arr = [1,2,3,4,4,4,7,7,7];

uniqArr(arr);
```



```
▼Set(5) {1, 2, 3, 4, 7}
  ▼[[Entries]]
    ▶0: 1
    ▶1: 2
    ▶2: 3
    ▶3: 4
    ▶4: 7
    size: (...)
  ▶__proto__: Set
```

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

```
function flatten(ary) {

    var ret = [];

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

        if(Array.isArray(ary[i])) {

            ret = ret.concat(flatten(ary[i]));

        } else {

            ret.push(ary[i]);

        }

    }

    return ret;

}

console.log(flatten([[0, 1], [2, 3], [4, 5, [6, 7, [8, [9, 10]]]]]));
```

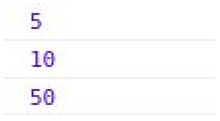


```
function flatten(ary) {
    var ret = [];
    for(var i = 0; i < ary.length; i++) {
        if(Array.isArray(ary[i])) {
            ret = ret.concat(flatten(ary[i]));
        } else {
            ret.push(ary[i]);
        }
    }
    return ret;
}

console.log(flatten([[0, 1], [2, 3], [4, 5, [6, 7, [8, [9, 10]]]]]));
▶ (11) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

8. Implement a singly linked list in es6 and implement addFirst() addLast(), length(), getFirst(), getLast(). (without using array)

```
class Node

{

    constructor(value)

    {

        this.value = value;

        this.next = null;

    }

}

class LinkedList{

    constructor(){

        this.root = null;

        this.size = 0;

    }

    addLast(value){

        const node = new Node(value);

        if(this.root)

        {

            let currentNode = this.root;

            while(currentNode && currentNode.next){

                currentNode = currentNode.next;

            }

            currentNode.next =  node;

        }

        else{
            this.root = node;
        }
        this.size++;
    }
    getlength()
    {
        return this.size;
    }
    getFirst()
    {
        return this.root.value;
    }
    getLast()
    {
        if(this.root)
        {
            let currentNode = this.root;
            while(currentNode && currentNode.next){
                currentNode = currentNode.next;
            }
            return currentNode.value;
        }
        else{
            return this.getFirst();
        }
    }
}
```

```
var ll = new LinkedList();

ll.addLast(10);

ll.addLast(20);

ll.addLast(30);

ll.addLast(40);

ll.addLast(50);

console.log(ll.getlength());

console.log(ll.getFirst());

console.log(ll.getLast());
```



```
5
10
50
```

9. Implement Map and Set using Es6?

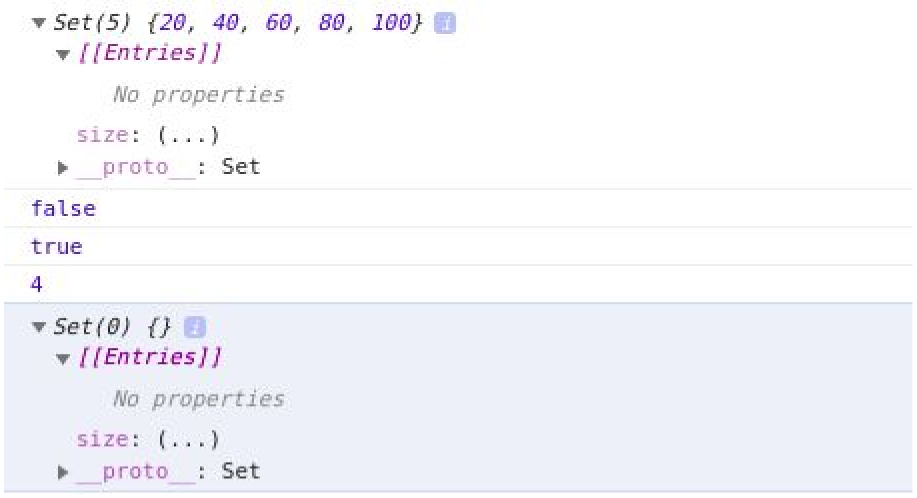
MAP

```
var myMap = new Map();

//adding value in map

//SET

myMap.set(1,'shivam');

myMap.set(2,'tarun');

myMap.set(3,'abhiman');

myMap.set(4,'vasu');

myMap.set(5,'pasha');

console.log(myMap);


//GET

console.log(myMap.get(1));
```

```
//HAS

console.log(myMap.has(6));

console.log(myMap.has(1));


//DELETE

myMap.delete(2);

console.log(myMap);


//SIZE

console.log(myMap.size);


//GET KEYS

console.log(myMap.keys());


//GET VALUES

console.log(myMap.values());


//CLEAR

console.log(myMap.clear());

console.log(myMap);
```

SET

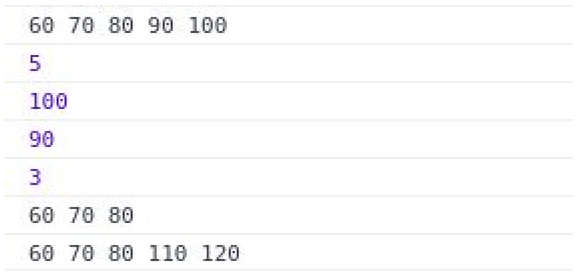
```
var mySet = new Set();

//adding values in set

mySet.add(20);

mySet.add(40);

mySet.add(60);

mySet.add(80);

mySet.add(100);

console.log(mySet);

//delete

mySet.delete(40);

//Has

console.log(mySet.has(40));

console.log(mySet.has(60));

//Size

console.log(mySet.size);

//Clear

mySet.clear();

console.log(mySet);
```



```
▼ Set(5) {20, 40, 60, 80, 100}
  ▼ [[Entries]]
      No properties
    size: (...)
  ▶ __proto__: Set
false
true
4
▼ Set(0) {}
  ▼ [[Entries]]
      No properties
    size: (...)
  ▶ __proto__: Set
```

10. Implementation of stack (using linked list) ?

```
class Node
{
    constructor(value)
    {
        this.value = value;
        this.next = null;
    }
}

class LinkedListStack {
    constructor() {
        this.root = null;
        this.size = 0;
    }

    addLast(value) {
        const node = new Node(value);
        if (this.root) {
            let currentNode = this.root;
            while (currentNode && currentNode.next) {
                currentNode = currentNode.next;
            }
            currentNode.next = node;
        }
        else {
            this.root = node;
        }
        this.size++;
        // console.log(value);
    }

    getlength() {
        return this.size;
    }

    removelast() {
        let val = null;
        if (this.root) {
            let currentNode = this.root;
            while (currentNode.next.next) {
                currentNode = currentNode.next;
            }
            val = currentNode.next.value;
            currentNode.next = null;
            this.size--;
            return val;
        }
        else {
            return -1;
        }


    }
    printList() {
        var curr = this.root;
        var str = "";
        while (curr) {
            str += curr.value + " ";
            curr = curr.next;
        }
        console.log(str);
    }

}

var ls = new LinkedListStack();
ls.addLast(60);
ls.addLast(70);
ls.addLast(80);
ls.addLast(90);
ls.addLast(100);
ls.printList();

console.log(ls.getlength());
console.log(ls.removelast());
console.log(ls.removelast());
console.log(ls.getlength());
ls.printList();

ls.addLast(110);
ls.addLast(120);

ls.printList();
```



```
60 70 80 90 100
5
100
90
3
60 70 80
60 70 80 110 120
```