1. Filter unique array members using Set.

```
const arr_q1 = [12, 23, 230, 3700, 12, 23, 34];
const arr_q1Set = new Set(arr_q1);
console.log(arr_q1Set); // Unique Members in a set
console.log(Array.from(arr_q1Set)); // Unique elements from the set
converted to array
```

```
▶ Set(5) {12, 23, 230, 3700, 34}
▶ (5) [12, 23, 230, 3700, 34]
```

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

```
const ques2 = 'ToTheNew';
const allCombinations = (ques2) => {
    var lenStr = ques2.length;
    var result = [];
    var indexCurrent = 0;

    while (indexCurrent < lenStr) {
        var char = ques2.charAt(indexCurrent);
        var x;
        var arrTemp = [char];

        for (x in result) {
            arrTemp.push("" + result[x] + char);
        }
        result = result.concat(arrTemp);

        indexCurrent++;
    }

    return result;
};
const arr_ques2 = allCombinations(ques2);
```

```
const map_ques2 = new Map();
arr_ques2.forEach(function (item, index) {
    map_ques2.set(index, item);
});
console.log(map_ques2);
```

```
▼Map(255) {0 => "T", 1 => "o", 2 => "To", 3 => "T", 4 => "TT", …}
  ▼[[Entries]]
    ▶[0 … 99]
    ▶[100 … 199]
    ▶[200 … 254]
    size: (...)
  ▶__proto__: Map
```

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

```
class Person {
    first;
    last;
    age;
    gender;
    constructor(first, last, age, gender) {
        this.name = {
            first,
            last
        };
        this.age = age;
        this.gender = gender;
    }

    static greeting(x) {
        console.log(`Hi! I'm ${x.name.first}`);
    };

    static farewell(x) {
        console.log(`${x.name.first} has left the building. Bye for
now!`);
    };
}
class Employee extends Person {
    id;
```

```
    constructor(first, last, age, gender, id) {
        super(first, last, age, gender);
        this.id = id;
    }
    static getID(x) {
        console.log(`${x.id} has checked in.`);
    };
}

class Developer extends Employee {
    position;
    constructor(first, last, age, gender, id, position) {
        super(first, last, age, gender, id);
        this.position = position;
    }
    static getPosition(x) {
        console.log(`${x.position} has checked out`);
    };
}
let p1 = new Person('Archit', 'Gupta', 22, 'M');
let e1 = new Employee('CK', 'Gupta', 58, 'M', 10121);
let d1 = new Developer('Preeti', 'Gupta', 55, 'F', 10122, 'MEAN');
```

```
p1
▼ Person {first: undefined, last: undefined, age: 22, gender: "M", name: {…}}
    age: 22
    first: undefined
    gender: "M"
    last: undefined
  ▶ name: {first: "Archit", last: "Gupta"}
  ▶ __proto__: Object

e1
▼ Employee {first: undefined, last: undefined, age: 58, gender: "M", name: {…}, …}
    age: 58
    first: undefined
    gender: "M"
    id: 10121
    last: undefined
  ▶ name: {first: "CK", last: "Gupta"}
  ▶ __proto__: Person
```

```
d1
▼Developer {first: undefined, last: undefined, age: 55, gender: "F", name: {…}, …}
   age: 55
   first: undefined
   gender: "F"
   id: 10122
   last: undefined
  ▶name: {first: "Preeti", last: "Gupta"}
   position: "MEAN"
  ▶__proto__: Employee
```

```
Employee.greeting
ƒ greeting(x) {
        console.log(`Hi! I'm ${x.name.first}`);
    }
Employee.greeting(e1);
Hi! I'm CK
```

```
Developer.getPosition(d1)
MEAN has checked out
undefined
Developer.getID(d1)
10122 has checked in.
undefined
```

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

```
class User {
    static names = [];

    static isNameTaken(name) {
        return User.names.includes(name);
    }

    name = 'Unknown';

    constructor(name) {
        this.name = name;
        User.names.push(name);
    }
}

const user = new User('Archit');
```

```
console.log(User.isNameTaken('Archit'));
console.log(User.isNameTaken('Raman'));
```

```
class User {
    static names = [];

    static isNameTaken(name) {
        return User.names.includes(name);
    }

    name = 'Unknown';

    constructor(name) {
        this.name = name;
        User.names.push(name);
    }
}

const user = new User('Archit');

console.log(User.isNameTaken('Archit'));
console.log(User.isNameTaken('Raman'));
```

```
true
false
```

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

```
function areaCircle(radius) {
    let area = Math.PI * Math.pow(radius, 2);
    // console.log('Area of the circle is ' + area);
    return area;
}

function areaRectangle(length, breadth) {
    let area = length * breadth;
    // console.log('Area of the rectangle is ' + area);
    return area;
}

function areaCylinder(radius, height) {
    let area = 2 * Math.PI * Math.pow(radius, 2) + (2 * Math.PI *
radius * height);
    // console.log('Area of the Cylinder is ' + a + ' square unit');
    return area;
}
```

```
export {
    areaCircle,
    areaCylinder,
    areaRectangle
}
```

```
import * as cal from './modules/area.js';
console.log(`Rectangle's Area = ${cal.areaRectangle(5, 6)}`);
console.log(`Cylinder's Area = ${cal.areaCylinder(5, 6)}`);
console.log(`Circle's Area = ${cal.areaCircle(7)}`);
```

```
Rectangle's Area = 30
Cylinder's Area = 345.5751918948772
Circle's Area = 153.93804002589985
```

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

```
const arr_q1 = [12, 23, 230, 3700, 12, 23, 34];
const arr_q1Set = new Set(arr_q1);
export { arr_q1, arr_q1Set };
```

```
import { arr_q1Set as result } from './modules/unique.js'
console.log(result);
```

```
▸ Set(5) {12, 23, 230, 3700, 34}
```

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

```
var ques7 = [[1, 2], 3, 4, [5, 6, 7, 8], [9, 11, 12]];
var myNewArray = ques7.reduce((prev, curr) => prev.concat(curr));
console.log(myNewArray);
```

```
▸ (11) [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12]
```

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

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

```
}

class LinkedList {
    constructor() {
        this.head = null;
        this.size = 0;
    }
}

LinkedList.prototype.addFirst = function (data) {
    let newNode = new Node(data);
    newNode.next = this.head;
    this.head = newNode;
    this.size++;
    return this.head;
}

LinkedList.prototype.addLast = function (data) {
    let newNode = new Node(data);
    if (!this.head) {
        this.head = newNode;
        return this.head;
    }
    let tail = this.head;
    while (tail.next !== null) {
        tail = tail.next;
    }
    tail.next = newNode;
    this.size++;
    return this.head;
}

LinkedList.prototype.getLast = function () {
    let lastNode = this.head;
    if (lastNode) {
        while (lastNode.next) {
            lastNode = lastNode.next
        }
    }
    return lastNode;
```

```
}

LinkedList.prototype.getFirst = function () {
    return this.head;
}

LinkedList.prototype.length = function () {
    return this.size;
}
```

```
let list = new LinkedList();
undefined
list.addFirst(2);
▶ Node {data: 2, next: null}
list.addLast(3);
▶ Node {data: 2, next: Node}
list.size
2
list.getLast
ƒ () {
    let lastNode = this.head;
    if (lastNode) {
        while (lastNode.next) {
            lastNode = lastNode.next
        }
    }
    return lastNode;
}
list.getLast()
▶ Node {data: 3, next: null}
list.getFirst()
▶ Node {data: 2, next: Node}
```

9. Implement Map and Set using Es6?

```
var ques9_map = new Map();
ques9_map.set(101, 1);
ques9_map.set(102, 2);
ques9_map.set(103, 3);
console.log(ques9_map);
console.log(ques9_map.get(102));
```

```
console.log(ques9_map.has(2));
console.log(ques9_map.size);
console.log(ques9_map.keys());
console.log(ques9_map.values());
console.log(ques9_map.entries());
console.log(ques9_map.delete(103));
console.log(ques9_map.clear());
console.log(ques9_map);

var ques9_set = new Set();
ques9_set.add(1001);
ques9_set.add(1002);
ques9_set.add(1003);
console.log(ques9_set);
console.log(ques9_set.delete(1003));
console.log(ques9_set);
console.log(ques9_set.has(1001));
console.log(ques9_set.size);
console.log(ques9_set.clear());
console.log(ques9_set);
```

```
▶ Map(3) {101 => 1, 102 => 2, 103 => 3}
2
false
3
▶ MapIterator {101, 102, 103}
▶ MapIterator {1, 2, 3}
▶ MapIterator {101 => 1, 102 => 2, 103 => 3}
true
undefined
▶ Map(0) {}
▶ Set(3) {1001, 1002, 1003}
true
▶ Set(2) {1001, 1002}
true
2
undefined
▶ Set(0) {}
```

10. Implementation of stack (using linked list) ?

```
class Stack {
    constructor() {
        this.first = null;
        this.last = null;
        this.size = 0;
    }
}

Stack.prototype.push = function (data) {
    let newNode = new Node(data);
    if (!this.first) {
        this.first = newNode;
        this.last = newNode;
    } else {
        let temp = this.first;
        this.first = newNode;
        this.first.next = temp;
    }
    return ++this.size;
}

Stack.prototype.pop = function (data) {
    if (!this.first) {
        return null;
    }

    let temp = this.first;
    if (this.first == this.last) {
        this.last = null;
    }
    this.first = this.first.next;
    this.size--;
    return temp.data;
}
```

```
let s = new Stack();
undefined
s.push(5);
1
s.push(6)
2
s.push(7)
3
s.pop(6);
7
s
▼ Stack {first: Node, last: Node, size: 2}
  ▶ first: Node {data: 6, next: Node}
  ▶ last: Node {data: 5, next: null}
    size: 2
  ▶ __proto__: Object
```