Map and Set

Map

A **map** is a collection of elements where each element is stored as a key, value pair. just like an Object. But the main difference is that Map allows keys of any type.

A **map** can also use objects as keys.

For example:

```
 let john = { name: "John" };
//Object created

let visitsCountMap = new Map();
//new map created

visitsCountMap.set(john, 123);
// john is the key for the map

alert( visitsCountMap.get(john) ); // 123
```

Using objects as keys is one of the most notable and important Map features.

Properties(methods) of Map :

- **Map.size**: It returns the number of elements or the key-value pairs in the map. `js  map1.size;`
- **Map.set()** : It takes two parameters key and value respectively and adds the key and value to the Map Object. `js  map1.set(6, 7);`
- **Map.has()** : It takes one parameter i.e the key and return a boolean value depending on whether the specified key is present or not `js  map1.has(1);`
- **Map.get()** : It takes one parameter i.e key and returns the value of corresponding key `js  map1.get(4);`
- **Map.delete()** : It takes key as a parameter and delete's both the key as well as a value from the map. `js  map1.delete(4);`
- **Map.clear()** : Removes all the elements from the Map object.

`js  map1.clear();`

- **Map.keys()** : It returns an iterator object which contains all the keys present in the Map Object. `js  map1.keys();`
- **Map.values()** : It returns an iterator object which contains all the values present in the Map Object. `js  map1.values();`

Example

```
 //creating a map
//map contains the data in  Key => Value format

var map1 = new Map([[1 , 2], [2 ,3 ] ,[4, 5]]);
console.log("Map1", map1); // Map {1=>2,2=>3,4=>5}

//performing different operations on map
// calculating size of map1

var size = map1.size;
console.log("Size of map1 is ", size); //3

// adding another pair to map
map1.set(3, 4); // here 3 is key and 4 is its value.
console.log("updated map is ", map1); //Map{1=>2,2=>3,4=>5,3=>4}

//checking whether map contains a key or not ?
var result = map1.has(2); // should return true as map1 has key = 2
console.log("Map contains 2 as key ", result);

var result2 = map1.has(7); // should return false as map1 does not have key = 7
console.log("Map contains 7 as key ", result2);

// Fetching the value of particular key
var value = map1.get(1); // should return the value of key = 1 i.e value = 2
console.log("value at key = 1 is ", value);

// deleting a particular key value pair
map1.delete(3); // will delete the key value pair where key = 3.
console.log("updated map is ", map1);

//fetching all the keys stored in Map
var keys = map1.keys(); // will return list of keys
console.log("key list is :", keys);

//fetching all the keys stored in Map
var values = map1.values(); // will return list of values
console.log("value list is :", values);

//delete all entries from Map
map1.clear();
console.log("Final map is ", map1);
```

Set

Sets are a new object type with ES6 that allows you to create collections of unique values. The values in a set can be either simple primitives like strings or integers, but more complex object types like object literals or arrays can also be part of a set.

Its main methods are:

- `new Set()` – creates the a new set.
- `set.add(value)` – adds a value, returns the set itself.
- `set.delete(value)` – removes the value, returns true if value existed at the moment of the call, otherwise false.
- `set.has(value)` – returns true if the value exists in the set, otherwise false.
- `set.clear()` – removes everything from the set.
- `set.size` – It returns the elements count in the set.

For example, we have visitors coming, and we'd like to remember everyone. But repeated visits should not lead to duplicates. A visitor must be "counted" only once.

Set is just the right thing for that:

```
 let set = new Set();

let john = { name: "John" };
let pete = { name: "Pete" };
let mary = { name: "Mary" };

// visits, some users come multiple times
set.add(john);
set.add(pete);
set.add(mary);
set.add(john);
set.add(mary);

// set keeps only unique values
alert( set.size ); // 3

for (let user of set) {
  alert(user.name); // John (then Pete and Mary)
}
```

The alternative to Set could be an array of users, and the code to check for duplicates on every insertion using `arr.find`. But the performance would be much worse because this method walks through the whole array checking every element. The set is much better optimized internally for uniqueness checks.

Iteration over Set

Just like arrays, We can loop over a set either with `for..of` or using `forEach`:

```
let set = new Set(["oranges", "apples", "bananas"]);

for (let value of set) alert(value);
```