==Map and Set:================================

Set:

A Set is a special type collection – “set of values” (without keys), where each value may occur only once.

Its main methods are:
----

new Set([iterable]) – creates the set, and if an iterable object is provided (usually an array), copies values from it into the 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 – is the elements count.

**The main feature is that repeated calls of set.add(value) with the same value don’t do anything.
       **That’s the reason why each value appears in a Set only once.

**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.

example:

```
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)
}
```

Iteration over Set::

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);

// the same with forEach:
set.forEach((value, valueAgain, set) => {
  alert(value);
});
```

1.Objects are used for storing keyed collections.
2.Arrays are used for storing ordered collections.

==Map:============================
================================

Map is a collection of keyed data items, just like an Object. But the main difference is that Map allows keys of any type.

Methods and properties are:
----

new Map() – creates the map.
map.set(key, value) – stores the value by the key.
map.get(key) – returns the value by the key, undefined if key doesn’t exist in map.
map.has(key) – returns true if the key exists, false otherwise.
map.delete(key) – removes the element (the key/value pair) by the key.
map.clear() – removes everything from the map.

map.size – returns the current element count.

eg:

```
let map = new Map();

map.set('1', 'str1');   // a string key
map.set(1, 'num1');     // a numeric key
map.set(true, 'bool1'); // a boolean key

// remember the regular Object? it would convert keys to string
// Map keeps the type, so these two are different:
alert( map.get(1)   ); // 'num1'
alert( map.get('1') ); // 'str1'

alert( map.size ); // 3
```

Map can also use objects as keys.

For instance:

```
                  let john = { name: "John" };

// for every user, let's store their visits count
let visitsCountMap = new Map();

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

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

Iteration over Map:
=======================

For looping over a map, there are 3 methods:

map.keys() – returns an iterable for keys,
map.values() – returns an iterable for values,
map.entries() – returns an iterable for entries [key, value], it’s used by default in for..of.

```
                  eg:
                  ----


let recipeMap = new Map([
  ['cucumber', 500],
  ['tomatoes', 350],
  ['onion',    50]
]);

// iterate over keys (vegetables)
for (let vegetable of recipeMap.keys()) {
  alert(vegetable); // cucumber, tomatoes, onion
}

// iterate over values (amounts)
for (let amount of recipeMap.values()) {
  alert(amount); // 500, 350, 50
}

// iterate over [key, value] entries
for (let entry of recipeMap) { // the same as of recipeMap.entries()
  alert(entry); // cucumber,500 (and so on)
}
```

Object.fromEntries: Object from Map:

We’ve just seen how to create Map from a plain object with Object.entries(obj).

There’s Object.fromEntries method that does the reverse: given an array of [key, value] pairs, it creates an object from them:

```
let prices = Object.fromEntries([
  ['banana', 1],
  ['orange', 2],
  ['meat', 4]
]);

// now prices = { banana: 1, orange: 2, meat: 4 }

alert(prices.orange); // 2
```