ES6 Map

- A Map object can be used to hold key/value pairs. A key or value in a map can be anything (objects and primitive values).
- The syntax new Map([iterable]) creates a Map object where iterable is an array or any other iterable object whose elements are arrays (with a key/value pair each).
- An Object is similar to Map but there are important differences that make using a Map preferable in certain cases:
- 1. The keys can be any type including functions, objects, and any primitive.
- 2. You can get the size of a Map.
- 3. You can directly iterate over Map.
- 4. Performance of the Map is better in scenarios involving frequent addition and removal of key/value pairs.
- The size property returns the number of key/value pairs in a map.
  For example:

```
let map = new Map([
  ["k1", "v1"],
  ["k2", "v2"],
]);

console.log(map.size); // 2
```

Methods

- set(key, value) Adds a specified key/value pair to the map. If the specified key already exists, value corresponding to it is replaced with the specified value.
- get(key) Gets the value corresponding to a specified key in the map. If the specified key doesn't exist, undefined is returned.
- has(key) Returns true if a specified key exists in the map and false otherwise.
- delete(key) Deletes the key/value pair with a specified key from the map and returns true. Returns false if the element does not exist.
- clear() Removes all key/value pairs from map.
- keys() Returns an Iterator of keys in the map for each element.
- values() Returns an Iterator of values in the map for each element.
- entries() Returns an Iterator of array[key, value] in the map for each element.

- For example:

```
let map = new Map();
map.set("k1", "v1").set("k2", "v2");

console.log(map.get("k1")); // v1
console.log(map.has("k2")); // true

for (let kv of map.entries()) console.log(kv[0] + " : " + kv[1]);
```

- The above example demonstrates some of the ES6 Map methods.
- Map supports different data types i.e. 1 and "1" are two different keys/values.

ES6 Set

- A Set object can be used to hold unique values (no repetitions are allowed).
- A value in a set can be anything (objects and primitive values).
- The syntax new Set([iterable]) creates a Set object where iterable is an array or any other iterable object of values.
- The size property returns the number of distinct values in a set.
- For example:

```
let set = new Set([1, 2, 4, 2, 59, 9, 4, 9, 1]);

console.log(set.size); // 5
```

Methods

-add(value) Adds a new element with the given value to the Set.
-delete(value) Deletes a specified value from the set.
-has(value) Returns true if a specified value exists in the set and false otherwise.
-clear() Clears the set.
-values() Returns an Iterator of values in the set.

-For example:

```
let set = new Set();
set.add(5).add(9).add(59).add(9);

console.log(set.has(9));

for (let v of set.values()) console.log(v);
```

- The above example demonstrates some of the ES6 Set methods.
- Set supports different data types i.e. 1 and "1" are two different values.
- NaN and undefined can also be stored in Set.