JavaScript Map & Set - Comprehensive Notes

---

1. Map - Key Concepts

Definition: - Map is a collection of **key-value pairs**. - Keys can be of **any type** (string, number, object, function). - Insertion order is **preserved**.

Key Points: - Duplicate keys **are not allowed**; inserting the same key updates the value. - Provides **fast lookup** (`O(1)` on average). - Can be **iterated** using `for..of`, `map.keys()`, `map.values()`, `map.entries()`. - Supports **size property**: `map.size`. - Can convert **Map → Array**: `[...map]` - Can convert **Map → Object**: `Object.fromEntries(map)`

Basic Operations:

```
let map = new Map();
map.set('name', 'Neha');    // Add key-value
map.get('name');             // Retrieve value
map.has('name');             // Check key existence
map.delete('name');          // Delete key
console.log(map.size);       // Size of Map
```

---

2. Set - Key Concepts

Definition: - Set is a collection of **unique values**. - Insertion order is **preserved**.

Key Points: - Automatically **removes duplicates**. - Fast lookup: `O(1)` average for `set.has(value)`. - Can be iterated using `for..of`. - Supports **size property**: `set.size`. - Can convert **Set → Array**: `[...set]` - Useful for **set operations**: union, intersection, difference.

Basic Operations:

```
let set = new Set();
set.add(1);                // Add value
set.has(1);                // Check value existence
set.delete(1);             // Delete value
console.log(set.size);    // Size of Set
```

Removing duplicates from Array:

```
let arr = [1,2,2,3];
let uniqueArr = [...new Set(arr)]; // [1,2,3]
```

---

3. Map vs Set - Quick Comparison

Feature	Map	Set
Stores	Key-Value Pairs	Unique Values
Duplicate	Keys unique	Values unique
Access	By key	Iterate over values
Order	Preserved	Preserved
Add	map.set(key,value)	set.add(value)
Delete	map.delete(key)	set.delete(value)
Size	map.size	set.size

---

4. Examples

```
// Map Example
let map = new Map([['name','Neha'], ['age',22]]);
console.log([...map]); // [['name','Neha'], ['age',22]]

// Set Example
let set = new Set([1,2,2,3]);
console.log([...set]); // [1,2,3]
```

---

5. Tips for Interviews

- Use **Map** when you need **key-value pairs**.
- Use **Set** when you need **unique values**.
- Remember that both **Map** and **Set** preserve insertion order.
- Map keys can be **any type**, while Set values are **unique only by reference for objects**.
- Know basic operations: `add`, `delete`, `has`, `size`, iteration.
- Conversion between **Map/Set ↔ Array/Object** is common in problems.

---

6. Interview Questions & Answers - Map & Set

Map Questions

Q1: What is a Map and how is it different from an Object?
A: A Map is a collection of key-value pairs where keys can be of any type. Objects, however, only allow string or symbol keys. Maps also maintain insertion order and have a built-in `size` property.

Q2: How can you add, retrieve, and delete key-value pairs?
A: - Add: `map.set(key, value)` - Retrieve: `map.get(key)` - Delete: `map.delete(key)`

Q3: Can keys in a Map be objects or functions?
A: Yes, Maps allow any data type as a key including objects and functions.

Q4: What happens if the same key is added twice?
A: The value is updated, not duplicated.

Q5: How do you iterate over a Map?
A: You can use `for..of`, `map.forEach()`, or `map.keys()`, `map.values()`, `map.entries()`.

Q6: How to convert Map to Array and Object?
A: - Map → Array: `[...map]` - Map → Object: `Object.fromEntries(map)`

Set Questions

Q1: What is a Set and how is it different from an Array?
A: A Set is a collection of unique values, whereas arrays can contain duplicates. Sets also provide faster lookup with `has()`.

Q2: How can you add, check, and delete values in a Set?
A: - Add: `set.add(value)` - Check: `set.has(value)` - Delete: `set.delete(value)`

Q3: How do you remove duplicates from an array using a Set?
A:

```
let arr = [1, 2, 2, 3];
let unique = [...new Set(arr)]; // [1,2,3]
```

Q4: How to perform Union, Intersection, and Difference using Sets?
A: - Union: `new Set([...setA, ...setB])` - Intersection: `new Set([...setA].filter(x => setB.has(x)))` - Difference: `new Set([...setA].filter(x => !setB.has(x)))`

Q5: Does a Set preserve insertion order?
A: Yes, insertion order is preserved.

Q6: Can a Set store objects with the same content?
A: No, because Sets compare objects by reference. Two different objects with the same content are treated as different.

Combined Questions

Q1: When should you use Map vs Set?
A: Use Map for key-value associations, and Set when you need a collection of unique values.

Q2: What is the time complexity for basic operations in Map and Set?
A: Add, delete, and lookup operations are `O(1)` on average. Iteration is `O(n)`.

Q3: How do Map and Set handle objects as keys/values?
A: They store objects **by reference**. Even if two objects have the same content, they are considered different unless they are the same reference.

End of Notes