JavaScript Methods Cheat Sheet

📌JavaScript Map Methods

Map: A collection of key-value pairs where keys can be any type and maintains insertion order.

Core Methods

1. set(key, value) - Adds or updates an entry - Returns the Map object (can chain calls)

```
let map = new Map();
map.set("name", "Uday");
map.set(1, "number");
map.set({ id: 101 }, "object key");
console.log(map);
```

2. get(key) - Retrieves the value for a key

```
console.log(map.get("name")); // "Uday"
console.log(map.get(2)); // undefined
```

3. has(key) - Checks if a key exists

```
console.log(map.has("name")); // true
console.log(map.has("age"));  // false
```

4. delete(key) - Removes a key-value pair

```
map.delete("name");
console.log(map.has("name")); // false
```

5. clear() - Removes all entries

```
map.clear();
console.log(map.size); // 0
```

6. size - Returns the number of entries

```
map.set("a", 1).set("b", 2);
console.log(map.size); // 2
```

Iteration Methods

7. keys() - Iterator of keys

```
for (let key of map.keys()) console.log(key);
```

8. values() - Iterator of values

```
for (let value of map.values()) console.log(value);
```

9. entries() - Iterator of [key,value] pairs

```
for (let [key,value] of map.entries()) console.log(key,value);
```

Shortcut:

```
for (let [k,v] of map) console.log(k,v);
```

10. forEach(callbackFn) - Loop through entries

```
map.forEach((value,key)=>console.log(`${key} => ${value}`));
```

Other Features

Iterability: Spread syntax & Array.from

```
console.log([...map]); // [['a',1],['b',2]]
console.log(Array.from(map.keys())); // ['a','b']
```

Object ↔ Map Conversion

```
let obj = { x: 10, y: 20 };
let m = new Map(Object.entries(obj));
let backToObj = Object.fromEntries(m);
```

📌JavaScript Object Methods

- Keys are strings/symbols, values can be any type.

Core Features

1. Object Creation

```
let obj1 = {};
let obj2 = new Object();
let obj3 = Object.create(null);
```

2. Property Access

```
let person = { name: "Uday", age: 25 };
console.log(person.name);   // dot notation
console.log(person['age']); // bracket notation
```

3. Add / Update / Delete Properties

```
person.city = "Hyderabad"; // add
person.age = 26;           // update
delete person.city;         // delete
```

Built-in Methods

Object.keys(obj)

```
console.log(Object.keys(person)); // ["name", "age"]
```

Object.values(obj)

```
console.log(Object.values(person)); // ["Uday", 26]
```

Object.entries(obj)

```
console.log(Object.entries(person)); // [["name","Uday"],["age",26]]
```

Object.fromEntries(arr)

```
let arr = [["x",10],["y",20]];
console.log(Object.fromEntries(arr)); // { x:10, y:20 }
```

📌JavaScript Array Methods

1. Creation & Basics

```
Array.isArray([1,2]);          // true
Array.from("abc");           // ['a','b','c']
Array.of(1,2,3);               // [1,2,3]
```

2. Adding / Removing Elements

```
let arr = [1,2,3];
arr.push(4);        // [1,2,3,4]
arr.pop();          // [1,2,3]
arr.unshift(0);     // [0,1,2,3]
arr.shift();        // [1,2,3]
arr.splice(1,1,9); // [1,9,3]
arr.slice(1,3);    // [9,3]
```

3. Searching

```
arr.indexOf(9);          // 1
arr.lastIndexOf(3);      // 2
arr.includes(3);         // true
arr.find(x=>x>1);       // 9
arr.findIndex(x=>x>1);  // 1
```

4. Iteration

```
arr.forEach(x=>console.log(x));
arr.map(x=>x*2);              // [2,18,6]
arr.filter(x=>x>1);           // [9,3]
arr.reduce((a,b)=>a+b,0);     // 18
arr.reduceRight((a,b)=>a+b,0);
arr.some(x>5);                 // true
arr.every(x>0);               // true
```

5. Sorting & Reordering

```
arr.sort();                   // [3,9,?]
arr.reverse();                // [?,...,3]
arr.flat(2);                  // flatten nested arrays
arr.flatMap(x=>[x,x*2]);      // map + flatten
```

6. Conversion

```
arr.join('-');                // "3-9-?"
arr.toString();               // "3,9,?"
arr.toLocaleString();         // locale string
```

7. Filling & Copying

```
arr.fill(0,1,3);            // fill indices 1 to 2 with 0
arr.copyWithin(0,2);         // copy part to another index
```

8. Checking Length

```
arr.length;                   // 3
```

🔗Highlighted Notes

- **Map:** Use for non-string keys or insertion order.
- **Object:** Use for string/symbol keys, JSON-friendly.
- **Array:** Use for ordered collections.
- **Mutating methods:** push, pop, shift, unshift, splice, sort, reverse, fill, copyWithin.
- **Non-mutating methods:** slice, map, filter, flat, flatMap, toSorted, toReversed, toSpliced, with.