✨LECTURE 16 — REDUCE, MAP & SET

🔷 REDUCE() — Array → Single Value

👉 What is reduce()?

`reduce()` ek **array method** hai jo poore array par iterate karke **ek final result** banata hai.

📌 Final result **kisi bhi type ka ho sakta hai**:

- 🔢 Number
- 🧵 String
- 📦 Object
- 📚 Array

💡 First-Principle Explanation (Box Analogy)

Socho ek **dabba (box)** hai:

- har array element dabbe me jaata hai
- dabba har step pe update hota rehta hai
- **end me jo bachta hai → wahi reduce ka output**

➡️ Isi process ko JS `reduce()` kehta hai.

🔷 Reduce Syntax

```
array.reduce((accumulator, currentValue) => {
  // logic
  return updatedAccumulator;
}, initialValue);
```

Harshal Chauhan

🔷 Parameters Explained

- **accumulator (acc)** → pichle step ka result
- **currentValue (curr)** → current element
- **initialValue** → starting value (MOST IMPORTANT)

📌 Rule:

- sum ke liye → `0`
- object banana ho → `{}`

🔷 Example 1️⃣: Sum of Numbers

```
const arr = [10, 20, 30, 40, 50];

const sum = arr.reduce((acc, curr) => acc + curr, 0);

console.log(sum);
```

✅ **Output**

```
150
```

🧠 **Working**

- acc = 0
- 0 + 10 → 10
- 10 + 20 → 30
- 30 + 30 → 60
- 60 + 40 → 100
- 100 + 50 → **150**

Harshal Chauhan

🔷 Example 2️⃣: Frequency Count (VERY IMPORTANT)

```
let fruits = [
  "orange","apple","banana",
  "orange","apple","banana",
  "orange","grapes"
];

const result = fruits.reduce((acc, curr) => {
  if (acc.hasOwnProperty(curr)) {
    acc[curr]++;
  } else {
    acc[curr] = 1;
  }
  return acc;
}, {});
```

✅ Output

```
{
  orange: 3,
  apple: 2,
  banana: 2,
  grapes: 1
}
```

📌 Use-cases

- word count
- votes count

Harshal Chauhan

- cart items
- analytics

🧠 **One-line Yaad Rakhna**

> **“Sabko mila, ek cheez bana” = reduce()**

🔷 MAP — Advanced Key-Value Store

👉 What is Map?

`Map` ek **built-in object** hai jo **key-value pairs** store karta hai.

🔷 Why Map is Powerful

- keys **kisi bhi type** ki ho sakti hain
- insertion order maintain hota hai
- fast add / delete / search

🔷 Basic Map Operations

```
const map1 = new Map();


map1.set(3, 90);

map1.set("Rohit", 45);

map1.set(20, "Mohan");


map1.set("Rohit", 40); // overwrite

map1.delete(3);


console.log(map1.has("Rohit")); // true

console.log(map1.size);         // 2
```

Harshal Chauhan

```
map1.clear();

console.log(map1); // Map(0) {}
```

🔷 Map with Initial Values

```
const map2 = new Map([

  [4, "rohit"],

  ["Moahn", "rohan"],

  [30, 9],

  [63, 78]

]);
```

🔷 Iterating Map

```
for (let [key, value] of map2) {

  console.log(key, value);

}


map2.forEach((value, key) => {

  console.log(key, "👉", value);

});
```

Harshal Chauhan

🔷 Map vs Object

Feature	Object	Map
Key Type	String	Any
Order	❌	✅
Size	❌	✅
Iterable	❌	✅

📌 Interview Points

- Map iterable hota hai
- `10 !== "10"` in Map

🔷 SET — Unique Values Collection

👉 What is Set?

`Set` sirf **unique values** store karta hai.
Duplicate values **automatically remove** ho jaati hain.

🔷 Set with Duplicates

```
const setA = new Set([10, 20, 30, 40, 10, 30]);

console.log(setA);
```

✅ Output

```
Set(4) {10, 20, 30, 40}
```

Harshal Chauhan

🔷 Set Methods

```
const set1 = new Set();


set1.add(4);

set1.add("Rohit");

set1.add(30);

set1.delete(6);


console.log(set1.size);
```

🔷 Array → Set → Array (Remove Duplicates)

```
let arr = [10, 30, 20, 10, 40, 50, 30];

arr = [...new Set(arr)];

console.log(arr);
```

✅ Output

```
[10, 30, 20, 40, 50]
```

📌 **MOST COMMON INTERVIEW QUESTION**

🔷 Set Operations

```
// Union

new Set([...setX, ...setY]);


// Intersection

new Set([...setX].filter(x => setY.has(x)));
```

Harshal Chauhan

🔁 SHORT SUMMARY – Ek Nazar Me

- ✅ **reduce()** → Array → single value/object
- ✅ **Map** → advanced key-value store
- ✅ **Set** → unique values only

🧠 Memory Trick

- reduce = calculation
- Map = lookup table
- Set = no duplicates

Harshal Chauhan