✨LECTURE 11 — OBJECTS IN JAVASCRIPT (PART-2)

📘 **JavaScript Core — Destructuring & Prototype System**
🧠 *Clean Syntax • Deep Internals • Interview Gold*

⬛ FIRST PRINCIPLE — "DESTRUCTURING KYU CHAHIYE?"

🔷 Problem

Object se baar-baar likhna:

```
obj.name

obj.age

obj.balance
```

👉 Code **long + messy** ho jaata hai

🔷 Solution

👉 **Destructuring** = values ko **direct variables** me nikaal lena

---

⬛ 1. OBJECT DESTRUCTURING

🔷 Definition

Destructuring ek syntax hai jisme:

> Object ki properties ko **direct variables** me unpack kar lete hain

---

Basic Object Destructuring

```
let obj = {

  name: "Sourav",

  money: 420,
  balance: 30,

  age: 20,

  aadhaar: "74729826543"

};
```

❌ **Normal Access**

```
let n = obj.name;

console.log(n); // Sourav
```

✅ **With Destructuring**

```
const { name, balance, age } = obj;

console.log(name, balance, age);

// Sourav 30 20
```

🧠 **Rule**

> Variable ka naam **key ke naam se match** hona chahiye

⬛ RENAMING VARIABLES (IMPORTANT)

```
const { name: fullname, age: years } = obj;

console.log(fullname, years);

// Sourav 20
```

⚠️ **Important**

> Ab `name` aur `age` directly available ❌
> Sirf `fullname` aur `years` use kar sakte ho

⬛ REST OPERATOR WITH OBJECT (`...`)

🔷 Concept

Jo properties destructure **nahi** hoti,
wo **rest operator** ek naye object me daal deta hai

```
const { name, age, ...obj1 } = obj;


console.log(name, age);

// Sourav 20


console.log(obj1);

// { money: 420, balance: 30, aadhaar: "74729826543" }
```

🧠 Step-by-Step Soch

- `name` → variable bana
- `age` → variable bana
- baaki sab → `obj1` me pack

⚠️ Note

Original object se properties **delete nahi hoti**
Sirf copy hoti hain

⬛ 2. ARRAY DESTRUCTURING ⬛

🔷 Same Concept, Different Structure

```
const arr = [3, 2, 1, 5, 10];
```

Basic

```
const [first, second] = arr;

console.log(first, second);

// 3 2
```

Skipping Values

```
const [a, b, , c] = arr;

console.log(a, b, c);

// 3 2 5
```

Rest Operator in Array

```
const [x, y, ...rest] = arr;

console.log(x, y);   // 3 2

console.log(rest);   // [1, 5, 10]
```

🧠 **Rule**

Order matters in array destructuring

⬛3. NESTED DESTRUCTURING ⬛

🔷 Nested Object Destructuring

```
let obj = {

  name: "Harshal",

  age: 20,

  aadhaar: "45863072",

  address: {

    pincode: 802113,

    city: "Varanasi",

    state: "UP"

  }

};
```

```
const { address: { city, pincode } } = obj;

console.log(city, pincode);

// Varanasi 802113
```

🧠 **Meaning**

Object ke andar object → direct andar tak access

🔷 Array Inside Object

```
let obj = {

  arr: [90, 40, 60, 80]

};


const { arr: [first] } = obj;

console.log(first);

// 90
```

🧠 **Explanation**

- `arr:` → obj ke andar ki property
- `[first]` → us array ka pehla element

⬛ 4. PROTOTYPE CHAINING (VERY IMPORTANT) ⬛

🔷 First Principle

JavaScript me har object **akela nahi hota**

👉 Har object ke saath ek **hidden link** hota hai
👉 Is link ko bolte hain `__proto__`

🔷 Example

```
let obj = {

  name: "Harshal",

  amount: 420,

  greet: function() {

    return 10;

  }

};


console.log(obj.toString());
```

🟨 **Output**

```
[object Object]
```

⚡ **Question**

Humne `toString()` likha hi nahi — fir kaise mila?

🧠 **Answer**

`Object.prototype` se inherit hua

⬛ PROTOTYPE SEARCH FLOW

1️⃣ JS pehle **object ke andar** dekhega
2️⃣ Nahi mila → `__proto__` me dekhega
3️⃣ Chain chalte-chalte
4️⃣ End hota hai → **null**

👉 Isi process ko **Prototype Chaining** kehte hain

⬛ OBJECT LINKING EXAMPLE

```
let user1 = { name: "Harsh", age: 20 };

let user2 = { amount: 150, money: 20 };


user2.__proto__ = user1;


console.log(user2.name);

// Harsh
```

🧠 **Meaning**

user2 me `name` nahi mila
JS user1 (prototype) me chala gaya

⚠️ **Note**

`__proto__` ek **hidden reference link** hai
Inheritance ke liye use hota hai

⬛ PROTOTYPE CHAIN IN ARRAYS

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


arr.__proto__ === Array.prototype         // true

arr.__proto__.__proto__ === Object.prototype // true

Object.prototype.__proto__ === null       // true
```

⬛ PROTOTYPE CHAIN DIAGRAM

```
null

 ↑

Object.prototype → toString(), valueOf()
```

```
 ↑
Array.prototype  → push(), pop(), includes()
 ↑
arr = [10, 20, 30]
```

⬛ KEY INTERVIEW POINTS

✅ Har object → `Object.prototype` se linked
✅ Har array → `Array.prototype` se linked
✅ Isi wajah se `typeof array === "object"`
✅ Prototype chaining se **methods inherit** hote hain
✅ Chain ka end hamesha **null** hota hai

⬛ FINAL SUMMARY — EK NAZAR ME ⬛

🔷 Destructuring → short & readable syntax
🔷 Object + Array dono me kaam karta hai
🔷 Rest operator → baaki values ko pack karta hai
🔷 Nested destructuring → deep access
🔷 Prototype chaining → inheritance system
🔷 `__proto__` → hidden linking mechanism
🔷 Array methods → Array.prototype se aate hain

⬛ FINAL THOUGHT ⬛

🧠 **JavaScript ka heart = Objects + Prototypes**
Ye samajh liya →
👉 **JS internals, interviews, frameworks sab clear**