✨⬛ LECTURE 3 — JAVASCRIPT NON-PRIMITIVE DATA TYPES & TYPE CONVERSION

📘 **JavaScript Fundamentals — Reference Types & Type Conversion**
🧠 *First Principles • Real-Life Mapping • Zero Confusion*

⬛ FIRST PRINCIPLE — "DATA EK SE ZYADA KYU HOTA HAI?"

🔷 Simple Real-Life Soch

Socho ek **list** banani hai:

- Shopping items
- Students ke records
- User ki details

👉 Har value ke liye alag-alag variable banana **inefficient** hota hai.

🟨 **Solution**

JavaScript me **Non-Primitive (Reference) Data Types** hote hain
jo **multiple values** ko ek saath handle kar sakte hain.

⬛ NON-PRIMITIVE DATA TYPES (REFERENCE TYPES) ⬛⬛

🟨 **Important Property**

Ye data **value se nahi**,
memory address (reference) se kaam karta hai.

---

① ARRAY

🔷 Definition

Array ek **ordered collection** hota hai values ka
jo **index (0-based)** se access hoti hain.

🔷 Need (Why Array?)

- Jab multiple values ek hi variable me store karni ho
- Same type ya mixed values ho sakti hain

🔷 Real-Life Analogy

Playlist 🎵
Har song ka apna **index number**

```
let arr = [10, 20, 30.2, "Harshal", "rohit"];

console.log(arr);         // [10, 20, 30.2, "Harshal", "rohit"]

console.log(typeof arr); // object
```

🟨 IMPORTANT NOTE

`typeof arr` → **"object"**
kyunki **Array internally JavaScript Object par based hota hai**

---

② OBJECT

🔷 Definition

Object ek **collection of key-value pairs** hota hai.
Har property = **key : value**

🔷 Need (Why Object?)

- Real-world entities represent karne ke liye
  (User, Product, Car, Account, etc.)

🔷 Real-Life Analogy

ID Card 🪪
Name, Age, Number — sab ek hi entity ka part

```
let obj = {

  user_name: "Harshal",

  account_number: 31242314213,

  balance: 420

};
```

```
console.log(obj);

console.log(typeof obj); // object
```

🟨 **HIGHLIGHT**

Object tab use hota hai
jab **data structured + meaningful** ho

---

③ FUNCTION

🔷 Definition

Function ek **reusable block of code** hota hai
jo ek **specific task** perform karta hai.

🔷 Need (Why Function?)

- Code repetition avoid karne ke liye
- Logic ko reuse karne ke liye

🔷 Real-Life Analogy

Remote ka button 📺
Button dabao → same kaam repeat hota hai

```
let fun = function() {

  console.log("Hello coder Army");

};

fun();                    // Hello coder Army

console.log(typeof fun);  // function
```

🟨 **SPECIAL CASE**

`typeof function` → **"function"**
(Technically function bhi object hota hai,
par JS ne special category di hai)

⬛⬛ TYPE CONVERSION IN JAVASCRIPT ⬛⬛

⬛ FIRST PRINCIPLE — “TYPE KYU CHANGE HOTA HAI?”

JavaScript ek **loosely typed language** hai.
Isliye kabhi-kabhi data ka type **automatic ya manual** change ho jata hai.

---

① IMPLICIT TYPE CONVERSION (TYPE COERCION)

🔷 Meaning

Type conversion jo **JavaScript khud** karta hai.

```
console.log("5" + 2);  // "52"

console.log("5" - 2);  // 3

console.log(true + 1); // 2
```

🟨 RULES

- `+` → string ko priority deta hai
- `-` → number banane ki koshish karta hai
- `true` → 1, `false` → 0

---

② EXPLICIT TYPE CONVERSION

👉 Jab **developer khud** convert karta hai
using `Number()`, `String()`, `Boolean()`

✅ 1. String → Number

```
let account_balance = "100";

let num = Number(account_balance);


console.log(typeof account_balance); // string

console.log(typeof num);             // number
```

❌ **Invalid Conversion**

```
let account = "100xs";

console.log(Number(account)); // NaN
```

🟨 **NaN = Not a Number**

✅ 2. Boolean → Number

```
console.log(Number(true));  // 1

console.log(Number(false)); // 0
```

✅ 3. Null → Number

```
console.log(Number(null)); // 0
```

🟨 **Reason**

null = empty value → treated as 0

✅ 4. Undefined → Number

```
let x;

console.log(Number(x)); // NaN
```

🟥 **Undefined = unknown → NaN**

✅ 5. Number → String

```
let ab = 20;

console.log(String(ab)); // "20"
```

✅ 6. Boolean → String

```
let ax = true;

console.log(String(ax)); // "true"
```

✅ 7. String → Boolean

```
console.log(Boolean("str")); // true

console.log(Boolean(""));    // false

console.log(Boolean(" "));   // true
```

🟨 **RULE**

Empty string = false
Non-empty string (even space) = true

🟫🟫🟫🟫 SHORT SUMMARY — KEY LEARNINGS

✅ Array, Object, Function → **Non-Primitive (Reference Types)**
✅ Arrays ka `typeof` → `"object"`
✅ Functions ka `typeof` → `"function"`

Type Conversion Rules

- `"100"` → 100
- `"100abc"` → NaN
- `true` → 1, `false` → 0
- `null` → 0
- `undefined` → NaN
- `"hello"` → true
- `""` → false
- `" "` → true

🟫🟫🟫 FINAL ONE-LINE THOUGHT 🟫🟫🟫

🧠 **Primitive = Value copy**
🧠 **Non-Primitive = Reference copy**
🧠 **JavaScript flexible hai — par rules samajhna zaruri hai**