✨ LECTURE 13 — CONDITIONS, LOOPS & SCOPE

📘 **JavaScript Control Flow & Scope**
🧠 *Decision • Repetition • Visibility*

⬛ FIRST PRINCIPLE — “CODE KAISE DECIDE KARTA HAI?”

Real life me:

- Agar age ≥ 18 → vote allow
- Jab tak task complete na ho → repeat
- Kuch cheeze sirf andar kaam kare → scope

👉 Isi logic ko code me laane ke liye:

- **Conditions**
- **Loops**
- **Scope**

⬛ 1️⃣ CONDITIONAL STATEMENTS (Decision Making)

🔷 Definition

Conditional Statements code ko ye decide karne dete hain
ki **kaunsa block execute hoga aur kaunsa nahi**

✅ if Statement

🔷 Use Case

Jab **sirf ek condition** check karni ho

Harshal Chauhan

```
const age = 20;


if (age >= 18) {
  console.log("You are eligible to vote");
}
```

🧠 **Rule**

Condition true hui → block execute
false hui → ignore

✅ if – else Statement

🔷 Definition

Jab condition ke **do possible outcomes** ho

```
const age = 16;


if (age >= 18) {
  console.log("You are eligible to vote");
} else {
  console.log("You are not eligible to vote");
}
```

🧠 **Real-life**

Agar paisa hai → movie
warna → ghar

✅ if – else if – else Ladder

🔷 Definition

Jab **multiple conditions** sequentially check karni ho

Harshal Chauhan

```
let age = 19;


if (age < 18) {
  console.log("KID");
} else if (age > 45) {
  console.log("OLD");
} else {
  console.log("ADULT");
}
```

🧠 **Rule**

JS top se bottom check karta hai
Jo pehle true hua → wahi execute

✅ Switch Statement

🔷 Definition

Jab **same variable** ko multiple fixed values se compare karna ho

```
switch (new Date().getDay()) {
  case 0: console.log("Sunday"); break;
  case 1: console.log("Monday"); break;
  case 2: console.log("Tuesday"); break;
  case 3: console.log("Wednesday"); break;
  case 4: console.log("Thursday"); break;
  case 5: console.log("Friday"); break;
  case 6: console.log("Saturday"); break;
  default: console.log("Invalid day");
}
```

Harshal Chauhan

📌 **Tip**

Multiple fixed values → **switch zyada readable**

⬛ 2 LOOPS IN JAVASCRIPT (Repetition)

🔷 Definition

Loops ka use ek hi code ko
baar-baar execute karne ke liye hota hai

✅ for Loop

🔷 Definition

Jab number of iterations **pehle se pata ho**

```
for (let i = 1; i <= 5; i++) {
  console.log("Hello World");
}
```

🧠 Structure

```
for (start; condition; update)
```

🔷 for Loop with Array

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

for (let i = 0; i < arr.length; i++) {
  console.log(arr[i]);
}
```

📌 Most common real-world use

Harshal Chauhan

✅ while Loop

🔷 Definition

Jab **condition based loop** chahiye
aur iterations fixed na ho

```
let i = 0;


while (i < 5) {
  console.log("Hello World");
  i++;
}
```

🧠 Rule

Pehle condition check hoti hai
phir body execute hoti hai

✅ do…while Loop

🔷 Definition

Loop jo **kam se kam 1 baar** zaroor chale

```
let i = 0;


do {
  console.log("Hello World");
  i++;
} while (i < 5);
```

📌 Difference

while → pehle check
do-while → pehle run

Harshal Chauhan

✅ Nested Loops

🔷 Definition

Loop ke andar loop
Mostly **2D arrays / matrices** ke liye

```
const matrix = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
];

for (let i = 0; i < matrix.length; i++) {
  for (let j = 0; j < matrix[i].length; j++) {
    console.log(matrix[i][j]);
  }
}
```

🧠 Real-life

Rows ke andar columns

⬛ 3 SCOPE IN JAVASCRIPT (Visibility Rule)

🔷 Definition

Scope decide karta hai
ki variable **kaha accessible hoga**

Harshal Chauhan

✅ Global Scope

🔷 Definition

Jo variables **pure program me available** hote hain

```
let a = 10;

var b = 20;

const c = 30;


function greet() {

  console.log(a, b, c);

}


greet();

console.log(a, b, c);
```

⚠ **Avoid too much global data** → bugs aate hain

✅ Local / Function Scope

🔷 Definition

Function ke andar declared variables
bahar access nahi hote

```
function greet() {

  let a = 10;

  var b = 20;

  const c = 30;

  console.log(a, b, c);

}


greet();
```

Harshal Chauhan

```
console.log(a); // ❌ Error
```

✅ Block Scope (let & const)

🔷 Definition

`{ }` ke andar limited access

```
if (true) {
  let a = 10;
  const c = 30;
  var b = 20;

  console.log(a, c);
}

console.log(b); // ✅ var leak
console.log(a); // ❌ Error
```

🧠 Golden Rule

let & const = block scope
var = function scope

❌ Why NOT Use `var`?

🔷 Problems

- Block scope follow nahi karta
- Redeclaration allowed
- Hoisting confusion

```
console.log(x); // undefined
```

Harshal Chauhan

```
var x = 10;
```

📌 **Best Practice**

Hamesha **let / const** use karo

⬛ 4 FUNCTIONS & HOISTING

✅ Function Declaration (Hoisted)

```
greet();


function greet() {
  console.log("Hello World");
}
```

🧠 **Reason**

Function declaration memory me pehle load ho jata hai

❌ Function Expression (Not Hoisted)

```
meet(); // ❌ Error


const meet = function () {
  console.log("Hello Meet");
};
```

📌 **Why?**

Variable hoist hota hai → value nahi

Harshal Chauhan

⬛ FINAL QUICK SUMMARY ⬛

✅ Conditions → if, if-else, ladder, switch
✅ Loops → for, while, do-while, nested
✅ Scope → global, function, block
✅ let & const → safe
✅ var → avoid
✅ Function hoisting → declaration yes, expression no

⬛ FINAL THOUGHT

🧠 **JavaScript ka control system**
= Conditions + Loops + Scope

👉 In teen cheezon pe grip aa gayi
to **logic building strong** ho jata hai 💪

Harshal Chauhan