📄 Session Notes – 17-09-2025

---

1. Completion of JS Variable Declaration

- In JavaScript, variables can be declared using **var, let, const**.
- Summary from last session:
  - `var` → function-scoped, hoisted, re-declarable (not recommended in modern JS).
  - `let` → block-scoped, re-assignable, safer.
  - `const` → block-scoped, must be initialized, cannot be re-assigned.

👉 Example:

```js
var a = 10;     // function scoped
let b = 20;     // block scoped
const c = 30;   // block scoped, constant
```

---

2. Intro to Working Memory, RAM & Why JS is Slower than C/C++

- **Working Memory / RAM:**
  - JS variables and objects are stored in memory during runtime.
  - Two main parts:
    - **Stack (Primitive data types)** → fast access, fixed size.
    - **Heap (Objects & Functions)** → dynamic memory allocation.
- **Why JavaScript is slower than C/C++:**
  1. **Interpreted Language:**
     - JS is interpreted (via engines like V8), while C/C++ is compiled directly into machine code.
  2. **Dynamic Typing:**
     - JS decides variable types at runtime (slower).
     - C/C++ variables are statically typed.
  3. **Memory Management:**
     - JS uses Garbage Collection (automatic but adds overhead).
     - C/C++ uses manual memory management (faster, but complex).

☞ Key Insight:
JavaScript trades **speed for flexibility & ease of use**, while C/C++ prioritizes **performance**.

---

3. Data Types Introduction

- In JavaScript, data types are divided into **Primitive** and **Non-Primitive**.
- **Primitives** → immutable, stored in stack, copied **by value**.
- **Non-Primitives** → mutable, stored in heap, copied **by reference**.

---

4. Primitive Data Types (Completed)

1. **String** → `"Hello World"`
2. **Number** → `10, 3.14`
3. **Boolean** → `true, false`
4. **Null** → `let x = null;`
5. **Undefined** → `let y; // undefined`
6. **Symbol (ES6)** → `let id = Symbol("id");`
7. **BigInt (ES11)** → `let big = 1234567890123456789 0n;`

👉 Example:

```js
let str = "Sky";
let num = 100;
let isTrue = false;
let emptyVal = null;
let notDefined;
let uniqueId = Symbol("id");
let bigNum = 9007199254740991n;
```

---

5. Intro to Non-Primitive Data Types – Objects

- **Objects** are collections of key-value pairs.
- Stored in **Heap memory**.
- Copied **by reference** (not by value).

👉 Example:

```js
let person = {
  name: "Rajesh",
  age: 24,
  isStudent: true
};

console.log(person.name);  // Dot notation
console.log(person["age"]); // Bracket notation
```

⚡ Difference from Primitive:

```js
let x = 10;
let y = x;
y = 20;
console.log(x); // 10 (copy by value)

let obj1 = { value: 10 };
let obj2 = obj1;
obj2.value = 20;
console.log(obj1.value); // 20 (copy by reference)
```

✅ Conclusion

- JS variables can be declared with `var`, `let`, `const` (prefer `let`/`const`).
- JS uses **RAM (stack & heap)** for variable storage.
- JS is **slower** than C/C++ because it's interpreted, dynamically typed, and uses garbage collection.
- Completed **Primitive Data Types** (7 types).
- Introduced **Objects** → foundation for Non-Primitives.

📌 Assignment

1. Implement JS Objects with different properties (string, number, boolean).
2. Demonstrate **copy by value vs copy by reference**.
3. Research more on:
    - Stack vs Heap in memory.
    - Garbage Collection in JS.
    - Why dynamic typing affects performance.