Day-5

Task-1 : JavaScript Basics: Variables, Data Types, and Operators

1. Variables

Variables are used to store data values in JavaScript.

- **Declaring Variables:**
  JavaScript provides three keywords for declaring variables:
  - `var`: Globally or function-scoped.
  - `let`: Block-scoped, introduced in ES6.
  - `const`: Block-scoped, for constants that cannot be reassigned.

```
var name = "John"; // global or function-scoped
let age = 25; // block-scoped
const PI = 3.14; // cannot be reassigned
```

2. Data Types

JavaScript has two main categories of data types:

- **Primitive Types:**
  - `String`: Text values (`"Hello"`, `'World'`).
  - `Number`: Numeric values (`10`, `3.14`).
  - `Boolean`: Logical values (`true`, `false`).
  - `Undefined`: A variable declared but not assigned a value.
  - `Null`: Represents an intentional absence of value.
  - `Symbol`: Unique and immutable identifiers (introduced in ES6).
  - `BigInt`: Large integers (introduced in ES2020).
- **Non-Primitive Type:**
  - `Object`: Key-value pairs (`{}`), arrays, functions, etc

```
let text = "Hello"; // String
let number = 42; // Number
let isTrue = true; // Boolean
let nothing = null; // Null
let notDefined; // Undefined
let uniqueID = Symbol("id"); // Symbol
let bigNumber = 1234567890123456789012345678901234567890n; // BigInt
```

3. Operators

Operators are used to perform operations on variables and values.

- **Arithmetic Operators:** +, –, *, /, %, **
- **Comparison Operators:** ==, ===, !=, !==, <, >, <=, >=
- **Logical Operators:** && (AND), || (OR), ! (NOT)
- **Assignment Operators:** =, +=, -=, *=, /=, %=
- **String Operators:** + (Concatenation)
- **Type Operators:** `typeof`, `instanceof`

```
let x = 10, y = 5;
console.log(x + y); // 15
console.log(x > y); // true
console.log(x !== y); // true
console.log(typeof x); // "number"
```

Task-2 : JavaScript Control Flow: Conditional Statements & Loops

1. Conditional Statements

Control the flow of code execution based on conditions.

- **if statement:** Executes code if a condition is true.

```
if (x > y) {
  console.log("x is greater than y");
}
```

- **if-else statement:** Executes different code for true or false conditions.

```
if (x > y) {
  console.log("x is greater");
} else {
  console.log("y is greater");
}
```

- **else if statement:** Tests multiple conditions.

```
if (x > y) {
  console.log("x is greater");
} else if (x === y) {
  console.log("x is equal to y");
} else {
  console.log("y is greater");
}
```

- **switch statement:** Executes code based on multiple values of an expression.

```
let day = "Monday";
switch (day) {
  case "Monday":
    console.log("Start of the week");
    break;
  case "Friday":
    console.log("End of the work week");
    break;
  default:
    console.log("Middle of the week");
}
```

2. Loops

Loops are used to execute a block of code repeatedly.

- **for loop:** Iterates through a block of code a specific number of times.

```
for (let i = 0; i < 5; i++) {
  console.log(i); // 0, 1, 2, 3, 4
}
```

- **while loop:** Executes a block of code as long as a condition is true.

```
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

- **do...while loop:** Executes a block of code at least once, and then repeats while the condition is true.

```
let i = 0;
do {
  console.log(i);
  i++;
} while (i < 5);
```

- **for...of loop:** Iterates over iterable objects (like arrays).

```
let numbers = [1, 2, 3];
for (let num of numbers) {
  console.log(num); // 1, 2, 3
}
```

- **for...in loop:** Iterates over properties of an object.

```
let person = { name: "John", age: 25 };
for (let key in person) {
  console.log(key, person[key]); // name John, age 25
}
```