LAB ASSIGNMENT-3

1. Explain the differences between var, let, and const with respect to scope and hoisting.
   ANS: **1. var**
   var is a JavaScript keyword used to declare variables that are **function-scoped**. This means the variable is accessible anywhere inside the function in which it is declared. var is **hoisted** to the top of its scope and initialized with undefined, so it can be used before its declaration without causing an error.
   2. let
   let is used to declare variables that are **block-scoped**, meaning they are only accessible within the block {} where they are defined (such as loops or conditional statements). let is **hoisted** but not initialized, so accessing it before declaration results in a *Reference Error* due to the temporal dead zone.
   3. const
   const is used to declare variables that are also **block-scoped**. Like let, it is **hoisted but not initialized**, and accessing it before declaration causes a *Reference Error*. Additionally, variables declared with const must be assigned a value at the time of declaration and cannot be reassigned later.

2. Describe the various Control Flow statements in JavaScript, specifically highlighting the difference between for, while, and do while loops.

ANS:

Control flow statements control the order in which the program's instructions are executed. In JavaScript, they are mainly used for **decision making**, **looping**, and **branching**.

---

1. Conditional (Decision-Making) Statements

These statements execute code based on a condition.

- **if statement** – Executes code when a condition is true.

- **if…else statement** – Provides an alternative block if the condition is false.
- **else if ladder** – Checks multiple conditions in sequence.
- **switch statement** – Selects execution paths based on the value of an expression.

---

2. Looping (Iteration) Statements

Looping statements are used to repeat a block of code until a condition is met.

a) for Loop

- Used when the number of iterations is **known in advance**.
- Has initialization, condition, and increment/decrement in one line.

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

b) while Loop

- Used when the number of iterations is **not known beforehand**.
- The condition is checked **before** executing the loop body.

```
let i = 1;
while (i <= 5) {
  console.log(i);
  i++;
}
```

c) do…while Loop

- Similar to while loop, but the loop body is executed **at least once**.
- The condition is checked **after** the loop body.

```
let i = 1;
do {
```

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

Difference Between for, while, and do…while Loops

Feature	for Loop	while Loop	do…while Loop
Condition check	Before loop	Before loop	After loop
Minimum execution	0 times	0 times	At least 1 time
Best used when	Iterations are known	Iterations are unknown	Loop must run at least once
Structure	Compact	Simple	Slightly different syntax

---

3. Jumping (Branching) Statements

These statements alter the normal flow of execution.

- **break** – Exits a loop or switch statement.
- **continue** – Skips the current iteration and moves to the next.
- **return** – Exits a function and returns a value.

3. What is the Document Object Model (DOM)? Explain how to select elements and modify their content using inner Text and inner HTML.
   ANS:
   The **Document Object Model (DOM)** is a programming interface that represents an HTML document as a **tree structure**. In this tree, every part of the webpage—such as elements, attributes, and text—is treated as an **object**.
   The DOM allows JavaScript to **access, manipulate, and update** the content, structure, and style of a webpage dynamically without reloading it.

Selecting Elements in the DOM

JavaScript provides several methods to select HTML elements:

1. **getElementById()**
   Selects an element using its id.
   document.getElementById("title");
2. **getElementsByClassName()**
   Selects elements using a class name (returns a collection).
   document.getElementsByClassName("box");
3. **getElementsByTagName()**
   Selects elements using tag name.
   document.getElementsByTagName("p");
4. **querySelector()**
   Selects the **first** element that matches a CSS selector.
   document.querySelector(".box");
5. **querySelectorAll()**
   Selects **all** elements that match a CSS selector.
   document.querySelectorAll("p");

Modifying Content Using innerText and innerHTML

1. innerText

- Used to **get or set the visible text** content of an element.
- Ignores HTML tags.
  document.getElementById("title").innerText = "Welcome to JavaScript Lab";
  Example:
  <h1 id="title"></h1>

---

2. innerHTML

- Used to **get or set the HTML content** inside an element.
- Can include HTML tags.
  document.getElementById("content").innerHTML = "This is bold text";
  Example:

Difference Between innerText and innerHTML

Feature	innerText	innerHTML
Content type	Text only	Text + HTML
HTML tags	Not rendered	Rendered
Security	Safer	Can cause security issues if misused
Use case	Display plain text	Insert formatted content