JavaScript Lab Manual

This manual provides a detailed, topic-wise breakdown of JavaScript essentials, suitable for lab exercises or structured classroom learning.

Week 1: Introduction & Basics

Lab 1: JavaScript Introduction

Description: JavaScript is a lightweight, interpreted programming language with object-oriented capabilities. It is commonly used for enhancing interactivity and dynamic behavior on web pages. JavaScript is an essential part of web development alongside HTML and CSS. It runs in the browser, allowing client-side scripts to interact with the user and control the browser.

Key Concepts:

- JavaScript can be embedded in the HTML document within `<script>` tags or referenced as an external file.
- Can be placed in the `<head>` or `<body>` sections of HTML.

Example:

```
<script>
  alert("Hello, JavaScript!");
</script>
```

Lab 2: JavaScript Output

Description: JavaScript provides multiple methods to output data to users or developers. These outputs can be used for debugging, content display, or user interaction.

- `alert()` : Displays a pop-up alert box.
- `document.write()` : Writes directly to the HTML document.

- `console.log()` : Outputs data to the browser console, typically for debugging.
- `innerHTML` : Inserts or changes content of an HTML element.

Example:

```
<p id="demo"></p>
<script>
  alert("Welcome");
  document.write("Writing to document<br>");
  console.log("Logging to console");
  document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

More Examples:

```
console.log("This is a log message");
document.write("<h2>This is a heading written to document</h2>");
document.getElementById("demo").innerText = "Updated Text";
```

Lab 3: JavaScript Syntax

Description: JavaScript syntax is the set of rules that define a correctly structured JavaScript program. Understanding syntax is fundamental for writing efficient code.

- Statements end with a semicolon (`;`).
- JavaScript is case-sensitive.
- Blocks of code are contained within curly braces `{}` .
- Use of variables, functions, and operators follows strict syntax patterns.

Example:

```
let x = 5;
```

```
let y = 10;
let z = x + y;
```

More Examples:

```
const greeting = "Hello";
let name = "Alice";
console.log(greeting + ", " + name);

if (name === "Alice") {
  console.log("Welcome Alice");
}
```

Week 2: Variables & Operators

Lab 4: JavaScript Variables

Description: Variables are containers for storing data values. JavaScript uses three types of variable declarations:

- `var` : Globally or function scoped.
- `let` : Block scoped, introduced in ES6.
- `const` : Block scoped, cannot be reassigned. Variables help in storing dynamic values and are the backbone of any logic or data manipulation.

Example:

```
let name = "Alice";
const age = 30;
var country = "USA";
```

More Examples:

```
var a = 5;
```

```
let b = 6;
const c = a + b;
console.log(c);
```

Lab 5: JavaScript Operators

Description: Operators are symbols used to perform operations on variables and values. Types include:

- Arithmetic: `+`, `-`, `*`, `/`, `%`
- Assignment: `=`, `+=`, `-=`, `*=`, `/=`
- Comparison: `==`, `===`, `!=`, `>`, `<`, `>=`, `<=`
- Logical: `&&`, `||`, `!`
- String: `+` (concatenation) Operators form the basis of expressions and conditions in JavaScript.

Example:

```
let a = 5, b = 2;
console.log(a + b);
console.log(a > b && b < 3);
```

More Examples:

```
let x = 10;
x += 5;
console.log(x); // 15

let result = (x === 15) ? "Yes" : "No";
console.log(result);
```

Week 3: Data Types & Functions

Lab 6: JavaScript Data Types

Description: JavaScript supports a variety of data types, which determine the kind of data a variable can hold. These include:

- Primitive Types: String, Number, Boolean, Null, Undefined, Symbol
- Composite Types: Object, Array, Function Understanding data types is essential for effective data manipulation.

Example:

```
let str = "Hello";
let num = 10;
let isTrue = true;
let arr = [1, 2, 3];
let obj = {name: "John", age: 25};
```

More Examples:

```
let a = null;            // null value
let b;                   // undefined value
let symbol = Symbol();   // unique symbol

function greet() {
  return "Hi!";
}
console.log(typeof greet); // function
```

Lab 7: JavaScript Functions

Description: Functions are reusable blocks of code designed to perform a specific task. Functions enhance modularity and maintainability in programs. They can take parameters and return results.

- Declared using `function` keyword.

- Called by name with parentheses.

Example:

```
function add(a, b) {
  return a + b;
}
console.log(add(3, 4));
```

More Examples:

```
function greetUser(name) {
  console.log("Hello, " + name);
}

greetUser("Bob");

function square(x) {
  return x * x;
}
console.log(square(5));
```

Week 4: Control Statements

Lab 8: JavaScript Conditions

Description: Conditional statements allow decision making in code based on conditions.

- `if` executes block if condition is true.
- `else if` checks additional condition.
- `else` executes if none of the conditions are true.
- Ternary (`? :`) offers a shorthand for simple `if-else` .

Example:

```
let age = 20;
if (age >= 18) {
  console.log("Adult");
} else {
  console.log("Minor");
}
```

More Examples:

```
let temp = 30;
if (temp > 35) {
  console.log("Too hot");
} else if (temp < 15) {
  console.log("Too cold");
} else {
  console.log("Just right");
}

let status = (age >= 18) ? "Eligible" : "Not eligible";
console.log(status);
```

Lab 9: JavaScript Loops

Description: Loops are used to execute the same block of code repeatedly until a specified condition is met.

- `for` : Loop with defined start/end/step.
- `while` : Loop with condition checked before block.
- `do...while` : Executes at least once.
- `break` exits loop, `continue` skips to next iteration.

Example:

```
for (let i = 0; i < 5; i++) {
  if (i == 3) continue;
  console.log(i);
}
```

More Examples:

```
let i = 0;
while (i < 3) {
  console.log("While loop", i);
  i++;
}

i = 0;
do {
  console.log("Do While loop", i);
  i++;
} while (i < 3);
```

Week 5: Arrays and Objects

Lab 10: JavaScript Arrays

Description: Arrays are used to store multiple values in a single variable. JavaScript arrays are dynamic and can hold elements of different data types.

- Indexed from 0
- Common methods: `push()`, `pop()`, `shift()`, `unshift()`, `length`, `forEach()`

Example:

```
```

```
let fruits = ["Apple", "Banana", "Orange"];
fruits.push("Mango");
console.log(fruits);
```

More Examples:

```
let numbers = [1, 2, 3, 4];
numbers.pop();
console.log(numbers); // [1, 2, 3]

numbers.unshift(0);
console.log(numbers); // [0, 1, 2, 3]
```

Lab 11: JavaScript Objects

Description: Objects represent real-world entities. They are collections of key-value pairs (properties and methods).

- Accessed using dot `.` or bracket `[]` notation
- Methods are functions stored as object properties

Example:

```
let person = {
  name: "Bob",
  age: 28,
  greet: function() { return "Hi " + this.name; }
};
console.log(person.greet());
```

More Examples:

```
let car = {
```

```
  brand: "Toyota",
  model: "Corolla",
  year: 2020
};
console.log(car["model"]);

car.year = 2021;
console.log(car);
```

Week 6: Events and DOM

Lab 12: JavaScript Events

Description: Events are actions that occur in the browser that JavaScript can respond to. Common events include:

- `onclick` : Triggered on click
- `onmouseover` : Triggered when mouse moves over an element
- `onchange` : Triggered when input value changes

Example:

```
<button onclick="showMsg()">Click Me</button>
<p id="msg"></p>
<script>
function showMsg() {
  document.getElementById("msg").innerText = "Hello!";
}
</script>
```

More Examples:

```
<input type="text" id="input" onchange="changed()">
<script>
```

```
function changed() {
  alert("Input changed");
}
</script>

<div onmouseover="hoverEffect()">Hover over me!</div>
<script>
function hoverEffect() {
  alert("Mouse is over the div");
}
</script>
```

```
<button onmouseover="this.innerText='Hovered!'" onmouseout="this.innerText='Hover me'">Hover me</button>
```

Lab 13: JavaScript DOM

Description: The Document Object Model (DOM) is an interface for accessing and manipulating HTML and XML documents. JavaScript can dynamically change the structure, style, and content of a web page using the DOM.

- `getElementById`, `querySelector`
- `innerHTML`, `style`, `classList`

Example:

```
<p id="demo">JavaScript DOM</p>
<script>
document.getElementById("demo").style.color = "blue";
</script>
```

More Examples:

```
<div id="box">Original</div>
<script>
let box = document.getElementById("box");
box.innerHTML = "Changed";
box.classList.add("highlight");
</script>

<p id="text">Initial</p>
<script>
document.querySelector("#text").textContent = "Updated using querySelector";
</script>
```