Learning Report on JavaScript RKIT#4

# Introduction to JavaScript

JavaScript (JS) is a programming as well as scripting language used to create interactive and dynamic content on websites. It allows developers to add functionality such as animations, form validation, pop-ups, and more to make web pages come to life.

## Why Use JavaScript?

JavaScript is essential for making web pages interactive. While HTML provides the structure and CSS handles the design, JavaScript adds behavior. It allows developers to respond to user actions, manipulate content dynamically, and communicate with servers.

## How to Include JavaScript

JavaScript can be included in a web page in two ways:

1. Inline: You can add JavaScript directly inside an HTML tag using the 'script' tag.

Example: `<script>alert('Hello, world!');</script>`

1. External File: JavaScript code can be written in a separate file and linked using the 'script' tag.

Example: `<script src='app.js'></script>`

## Difference Between Loading in Head and End of Body

1. In Head: If you load JavaScript in the head, it will run before the page content is fully loaded. This can slow down the page load time. Example: `<head><script src='app.js'></script></head>`

2. End of Body: Loading JavaScript at the end of the body ensures that the HTML content loads first, making the page faster. Example: `<body><script src='app.js'></script></body>`

## Defer and Async Keywords

1. Defer: The 'defer' keyword delays the execution of JavaScript until after the HTML document is fully parsed. This helps avoid blocking the page load.

Example: `<script src='app.js' defer></script>`

2. Async: The 'async' keyword runs the script asynchronously, meaning it loads in parallel with other resources. However, it can run before the HTML document is fully parsed, potentially leading to issues.

Example: `<script src='app.js' async></script>`

# Types of Variables in JavaScript

1. var: Used to declare variables globally it is module scoped

Example: `var x = 10;`

2. let: Introduced in ES6, 'let' is used to declare variables with block scope (within curly braces).

Example: `let x = 10;`

3. const: Also introduced in ES6, 'const' is used to declare variables that cannot be reassigned. It also has block scope.

Example: `const x = 10;`

# JavaScript Operators

JavaScript uses different types of operators to perform actions on values and variables. Some common ones include:

1. Arithmetic Operators: Used for mathematical operations (e.g., +, -, \*, /).

Example: `let result = 10 + 5;`

2. Comparison Operators: Used to compare values (e.g., ==, ===, <, >).

Example: `if (x === 10) {}`

3. Logical Operators: Used to combine conditions (e.g., &&, ||, !).

Example: `if (x > 5 && y < 10) {}`

# JavaScript Events

JavaScript allows you to respond to events such as clicks, keypresses, and mouse movements. Some common events include:

1. Click Event: Triggers when an element is clicked.

Example: `element.addEventListener('click', function() {});`

2. Keypress Event: Triggers when a key is pressed.

Example: `element.addEventListener('keypress', function() {});`

3. Mouseover Event: Triggers when the mouse pointer hovers over an element.

Example: `element.addEventListener('mouseover', function() {});`

# Event Bubbling and Capturing

1. Event Bubbling: Events bubble up from the innermost element to the outermost. For example, if you click a button inside a div, the click event will first be handled by the button, then by the div.

2. Event Capturing: The opposite of bubbling. Events are captured from the outermost element to the innermost.

# Form Validation in JavaScript

JavaScript can be used to validate forms, ensuring that users enter required fields or proper data before submitting.

Example: You can check if a field is empty before allowing the form to be submitted.

<form onsubmit="return validateForm()"><input type="text" id="name" required><input type="submit"></form>