# DAY 9 (25-08-25)

# 1. Introduction to JavaScript

* JavaScript is a versatile programming language that's one of the core technologies of the World Wide Web, alongside HTML and CSS, and is used to create dynamic content for websites
* It is a lightweight, cross-platform, and single-threaded, interpreted and dynamically typed programming language.
* **Creator:** Brendan Eich
* **Initial Purpose:** To add simple **interactivity** to the Netscape browser.
* **Modern Importance:** It's an **essential core technology** for modern web application development.

## Key Features of JavaScript

Here are some key features of JavaScript that make it a powerful language for web development:

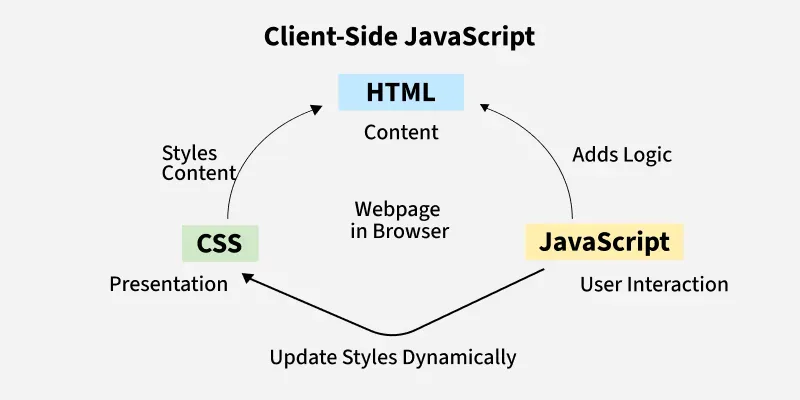
* **Client-Side Scripting:** JavaScript runs on the user's browser, so has a faster response time without needing to communicate with the server.
* **Versatile:** Can be used for a wide range of tasks, from simple calculations to complex server-side applications.
* **Event-Driven:** Responds to user actions (clicks, keystrokes) in real-time.
* **Asynchronous:** It can handle tasks like fetching data fromservers without freezing the user interface.
* **Rich Ecosystem:** There are numerous libraries and frameworks built on JavaScript, such as react, angular and Vue.js, which make development faster and more efficient.



## Where Does JavaScript Run?

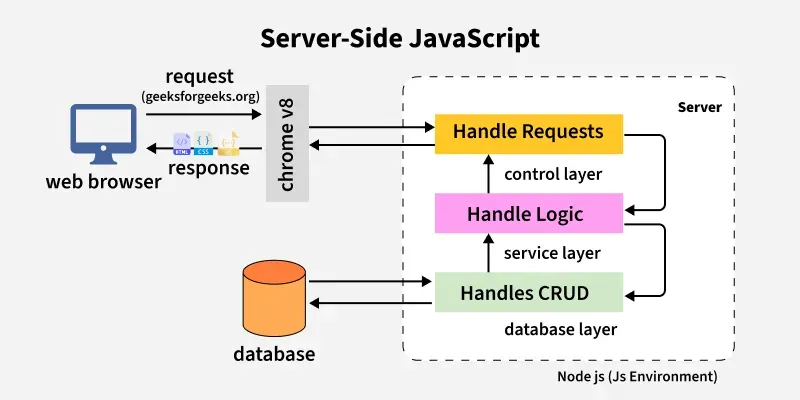
**Client-Side Scripting:**

* Involves controlling the browser and its DOM (Document Object Model).
* Handles user events like clicks and form inputs.
* Common libraries include AngularJS, ReactJS, and VueJS.



**Server-Side Scripting:**

* Involves interacting with databases, manipulating files, and generating responses.
* Node.js and frameworks like Express.js are widely used for server-side JavaScript, enabling full-stack development.



**Interaction between client and server**

## 

## 2. Basics of JavaScript

### Comments in JavaScript

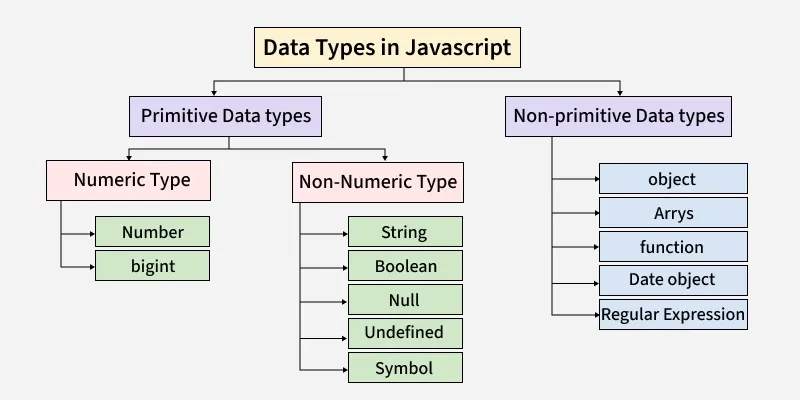
* **Single-line comment**: Starts with //
* **Multi-line comment**: Enclosed in /\* \*/

### Variables

Variables are containers for storing data values. JavaScript has three main keywords for declaring variables:

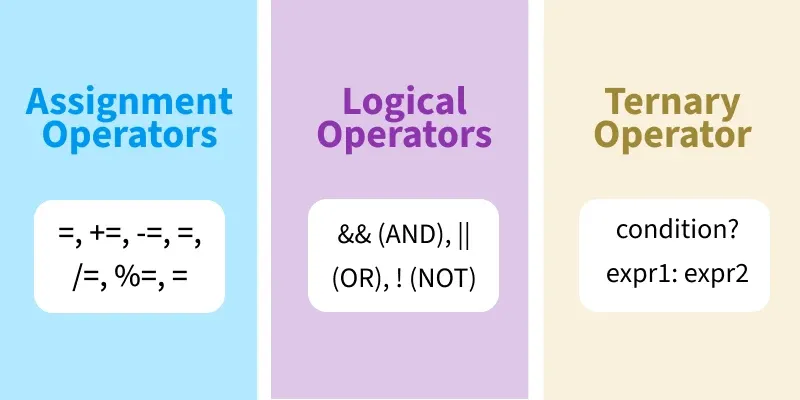
* **Var:** 
  + Has function scope.
* **let:**
  + **Block-scoped** (variables are accessible only within the block they are defined).
  + Can be reassigned but not redeclared in the same scope.
* **const:**
  + **Block-scoped**.
  + Cannot be redeclared or reassigned after initial assignment.
  + **Important:** For arrays and objects declared with const, the *reference* cannot be changed, but the contents (elements or properties) can still be modified.

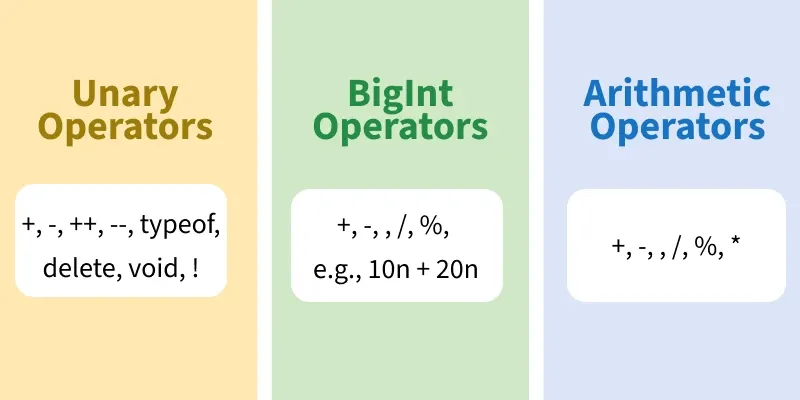
### Datatypes

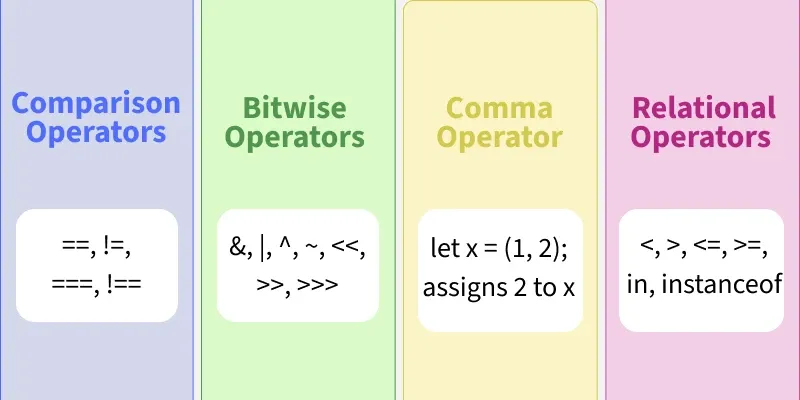


* **String:** Textual data (e.g., "Apexaiq", "Hello World").
* **Number:** Numeric data, including integers and floating-point numbers (e.g., 25, 3.14).
* **Boolean:** Logical values: true or false.
* **Null:** Represents the **intentional absence** of any object value. It's a primitive value.
* **Undefined:** Indicates that a variable has been declared but **has not yet been assigned a value**.
* **Object:** A complex data type that is a collection of **key-value pairs**.
* **Array:** An ordered collection of different types of values.

### Operators







## 3. Control Flow

Control flow statements determine the order in which instructions are executed.

### Conditional Statements

These statements execute different blocks of code based on whether a condition is true or false.

* **if:** Executes a block of code if a specified condition is true.
* **else:** Executes a block of code if the if condition is false.
* **else if:** Specifies a new condition to test if the preceding if and else if conditions are false.
* **switch:** Used to perform different actions based on different conditions, providing a more structured way to handle multiple else if scenarios.

### Loops

Loops are used to repeatedly execute a block of code.

* **for:** Loops through a block of code a specified number of times.
* **for/in:** Iterates over the **properties of an object**.
* **for/of:** Iterates over the **values of any iterable** object (like arrays, strings, maps, sets).
* **while:** Loops through a block of code as long as a specified condition is true.
* **do/while:** Similar to while, but it executes the block of code **at least once** before checking the condition.

## 4. Functions

### What is a Function?

A function is a **block of code** designed to perform a particular task. It runs only when it is called or invoked, promoting **code reusability**.

### Key Points About Functions

* **Parameters:** These are the **input values** that a function can accept.
* **Return:** This is the **output value** that a function can produce.
* **Scope:**
  + **Global Scope:** Variables declared globally are accessible from anywhere in the code.
  + **Local Scope:** Variables declared inside a function are only accessible within that function.
  + **Block Scope:** Variables declared with let or const inside {} blocks are only accessible within that block.

### Types of Functions

JavaScript offers various ways to define functions:

1. **Function Declaration (Named Function):**
   * Defined using the function keyword, usually with a name.
   * **Hoisted:** Can be called before its declaration in the code.

function greet(name) {  
 return `Hello, ${name}!`;  
}

1. **Function Expression:**
   * A function stored in a variable.
   * **Not hoisted:** Can only be used after its definition.

const sayHello = function(name) {  
 return `Hello, ${name}!`;  
};

1. **Arrow Function (ES6):**
   * A shorter syntax for writing function expressions using =>.
   * **Does not have its own this context**, which is important when dealing with objects and classes.

const greetArrow = (name) => `Hello, ${name}!`;

1. **Anonymous Function:**
   * A function without a name. Often used as **callback functions** or in IIFEs.

setTimeout(function() {  
 console.log("This runs after 1 second");  
}, 1000);

1. **Immediately Invoked Function Expression (IIFE):**
   * A function that runs **automatically** as soon as it's defined.
   * Used to create a private scope.

(function() {  
 console.log("This runs immediately!");  
})();

1. **Higher-Order Function:**
   * A function that either **takes another function as an argument** or **returns a function**.
   * Examples: map, filter, reduce.
2. **Recursive Function:**
   * A function that **calls itself** until a base condition is met.

## 5. Arrays and Objects

### Arrays

* **What is an Array?** An ordered list of values, acting as a container for multiple items.
* **Content:** Can store numbers, strings, objects, functions, and even other arrays.

#### Array Properties & Methods

* .length: Returns the total number of items in the array.
* .push(): Adds one or more items to the **end** of an array.
* .pop(): Removes the **last** item from an array.
* .shift(): Removes the **first** item from an array.
* .unshift(): Adds one or more items to the **start** of an array.
* .map(): Creates a **new array** by calling a provided function on every element in the calling array.
* .filter(): Creates a **new array** with all elements that pass the test implemented by the provided function.
* .reduce(): Executes a reducer function on each element of the array, resulting in a **single output value**.

### Objects

* **What are Objects?** A collection of **key-value pairs**.
* **Keys (Properties):** Always **strings or symbols**. They act as identifiers for the values.
* **Values:** Can be anything (string, number, array, function, object, etc.).

#### Features Provided by Objects

* **Accessing Properties:** Retrieving the value associated with a key (e.g., object.key or object['key']).
* **Adding / Updating Properties:** Assigning a new value to an existing key or a new key.
* **Deleting Properties:** Removing a key-value pair from an object (e.g., delete object.key).
* **Methods:** Functions defined as properties within an object.

## 6. DOM Manipulation

### What is the DOM?

The **Document Object Model (DOM)** is a **programming interface** for HTML and XML documents. It represents the page structure as a **tree of nodes**, where each node corresponds to an element, attribute, or text content in the document.

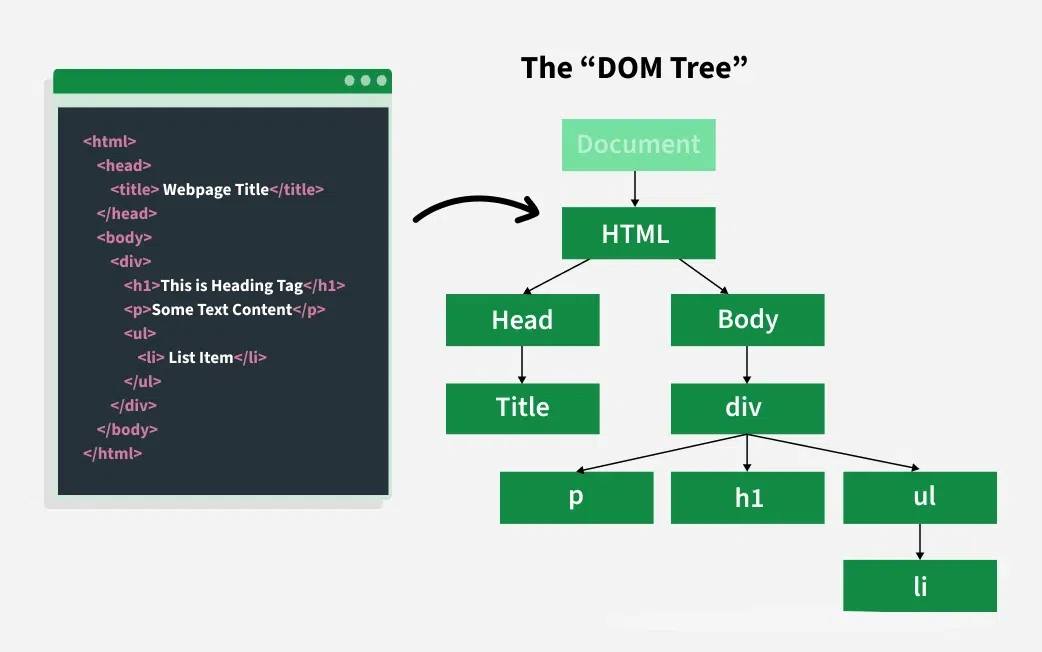
### Why is the DOM Important?

* **Dynamic Content:** Allows JavaScript to change page content (text, images) dynamically.
* **User Interaction:** Enables interaction with users (e.g., responding to button clicks, form submissions).
* **Dynamic Websites:** Crucial for creating interactive and dynamic web pages, moving beyond static content.

### DOM Manipulation Methods

JavaScript provides various methods to interact with and change the DOM:

* **Selecting Elements:** Finding specific HTML elements on the page.
  + document.getElementById('idName')
  + document.querySelector('.className') or document.querySelector('tagName')
  + document.querySelectorAll('tagName') (returns a NodeList)
* **Changing Content:** Modifying the text or HTML inside elements.
  + element.textContent = 'new text'
  + element.innerHTML = '<em>new HTML</em>'
* **Changing Style:** Altering the CSS properties of elements.
  + element.style.color = 'blue'
  + element.classList.add('new-class')
* **Creating New Elements:** Adding new HTML elements to the page.
  + document.createElement('div')
  + parentElement.appendChild(newElement)



## 7. Event Handling

### What is an Event?

An event is an **action that happens in the browser** (or to an element within it). These actions can be initiated by the user or the browser itself. JavaScript can "**listen**" for these events and execute specific code in response.

### Common Types of Events

* **onclick:** Triggered when a user **clicks** an element.
* **onmouseover:** Triggered when the mouse pointer is **moved over** an element.
* **onkeydown / onkeyup:** Triggered when a keyboard **key is pressed down** or **released**, respectively.
* **onsubmit:** Triggered when a **form is submitted**.
* **onchange:** Triggered when the **value of an input element changes**.

## Limitations of JavaScript

Despite its power, JavaScript has some limitations to consider:

* **Security Risks**: Can be used for attacks like Cross-Site Scripting (XSS), where malicious scripts are injected into a website to steal data by exploiting elements like <img>, <object>, or <script> tags.
* **Performance** : Slower than traditional languages for complex tasks, but for simple tasks in a browser, performance is usually not a major issue.
* **Complexity** : To write advanced JavaScript, programmers need to understand core programming concepts, objects, and both client- and server-side scripting, which can be challenging.
* **Weak Error Handling and Type Checking**: Weakly typed, meaning variables don’t require explicit types. This can lead to issues as type checking is not strictly enforced