The Dynamic Web: Introduction to JavaScript and the DOM

Welcome to Week 4! This presentation is your friendly guide into the essential world of JavaScript (JS) and the Document Object Model (DOM). These are the tools that transform static web pages into interactive, dynamic experiences.

By the end of this session, you will understand how to use JS to manipulate HTML and CSS, bringing your web applications to life. We will break down complex concepts—like objects, methods, and properties—using clear, simple language and real-world examples.

> JavaScript is the programming language of the web, and the DOM is its blueprint for interaction. Master these, and you master modern web development.

Chapter 1: Introducing JavaScript

Learning Goal:

Understand what JavaScript is, why it's used, and how to include it in an HTML page.

What is JavaScript?

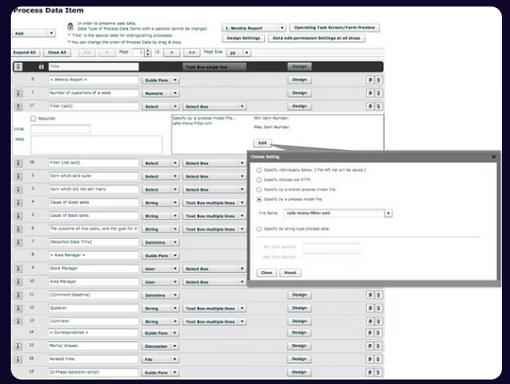
JavaScript (JS) is a high-level, interpreted programming language primarily used to add interactivity to websites. Think of HTML as the structure (the nouns), CSS as the style (the adjectives), and JavaScript as the behavior (the verbs).

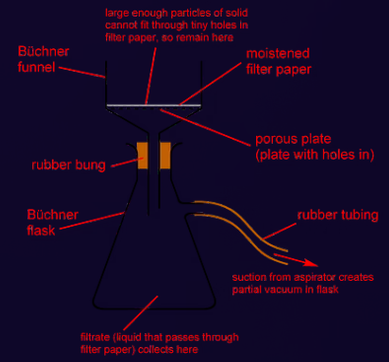
It allows you to implement complex features on web pages—everything from simple click responses to animated graphics, interactive maps, and modern web applications.

Where Does it Run?

Originally, JS was only run in the web browser (client-side). Now, thanks to technologies like Node.js, it can also run on the server (server-side), making it a full-stack language.

- Client-Side: Handles user interactions, updates page content without refreshing.
- Server-Side: Powers backend logic, database interactions, and API development.





How to Include JavaScript

There are two main ways to include JavaScript in your HTML document. For best practices, we almost always use external files to keep our code clean and separate from the structure.

1. Inline Scripting (Less Common)

This involves placing JS code directly inside the HTML using the `<script>` tags. This is often used for small, simple functions, but it can make your HTML messy.

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

It's important to place internal scripts strategically. Often, putting them just before the closing `</body>` tag ensures that the HTML content is loaded before the script tries to interact with it.

2. External Files (Best Practice)

This is the preferred method for any serious development. You write your JS code in a separate file (e.g., `script.js`) and link it to your HTML using the `src` attribute in the `<script>` tag.

```
<!-- In index.html -->
<body>
  <h1>My Site</h1>
  <script src="script.js"></script>
</body>
```

Using the `defer` or `async` attributes can further optimize loading, ensuring your page loads quickly while JS runs in the background or after HTML parsing.

The key takeaway is separation of concerns: HTML for structure, CSS for presentation, and JavaScript for behavior.

Chapter 2: Understanding the Document Object Model (DOM)

What is the DOM?

The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It represents the page so that programs can change the document structure, style, and content.

Imagine your HTML document as a family tree. The DOM is the structure that turns that HTML document into a set of connected "nodes" (objects). Every element, attribute, and piece of text in your HTML becomes a node in the DOM tree.

Interface

The DOM provides a standard way (an interface) for JavaScript to interact with HTML documents.

Tree Structure

It organizes all HTML elements into a logical tree structure, starting with the root `document` object.

Manipulation

JS uses the DOM to change, add, or remove elements, text, or attributes, which is how we create dynamic effects.

DOM: The Hierarchy of Nodes

To interact with the DOM, you must understand the different types of nodes and their relationships within the tree structure.

1

The Document Node (Root)

This is the entry point to all content on the page. Everything starts here. When you use `document.` in your JavaScript, you are accessing the top-level DOM object.

2

Element Nodes

These represent HTML tags, such as `<p>`, `<div>`, `<a>`, etc. These are the nodes we manipulate most often (changing text, adding classes, etc.).

3

Text Nodes

These contain the actual text content inside an Element Node. For example, the words inside a paragraph tag.

4

Attribute Nodes

These represent attributes of HTML elements, like `src` for an image or `href` for a link. While they are part of the DOM, they are typically treated as properties of their parent element node.

Remember: If you want to change something on the webpage, you need JavaScript to find the corresponding DOM node first!

Chapter 3: DOM Objects, Properties, and Methods

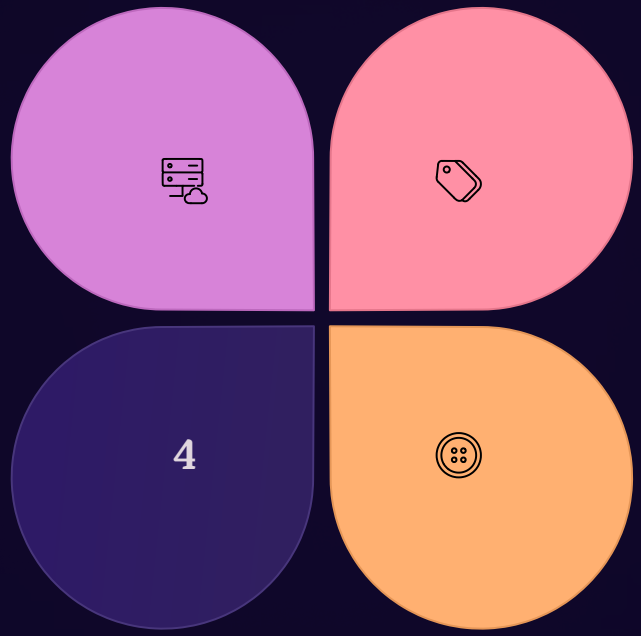
The DOM presents HTML elements as objects. These objects have properties (data/values) and methods (actions they can perform). This is the core vocabulary for interactive programming.

Objects

The basic building blocks. An object is a collection of related data and functionality (e.g., the browser window, the document, an HTML element).

Example Context

If a paragraph is an **Object**, its text content is a **Property**, and the ability to hide itself is a **Method**.



Properties

Characteristics or values associated with an object. Think of them as variables attached to an object (e.g., `element.id`, `element.value`).

Methods

Actions that can be performed on an object. They are functions that "belong" to an object (e.g., `document.getElementById()`, `element.addEventListener()`).

These concepts allow us to treat HTML elements as programmable entities.

Essential DOM Methods for Finding Elements

Before you can change an element, you must first tell JavaScript how to find it in the DOM tree. These are the most common methods for accessing nodes.

getElementById()	A single Element object	Finds one unique element using its `id` attribute. Since IDs must be unique, this is fast and precise.
getElementsByClassName()	A collection of elements	Finds all elements that share a specific CSS class. Useful for selecting multiple related items.
getElementsByTagName()	A collection of elements	Finds all elements of a specific HTML tag, such as all `<p>` or all `<img>` tags.
querySelector()	The first matching element	Finds the first element that matches a specified CSS selector (e.g., `#myID`, `.myClass`, `div p`). Modern and highly flexible.
querySelectorAll()	A list of all matching elements	Finds **all** elements that match a specified CSS selector. This is often preferred over the older collection methods.

The `querySelector` family is the modern standard because it uses the same powerful syntax as CSS selectors, making your selection logic easy to read and understand.

Code Example: Manipulating Content and Style

This example demonstrates how to find an element, change its content (`innerText` property), and change its style (`style` property) using DOM methods.

The HTML Structure

```
<h2 id="title">Initial Greeting</h2>
<button onclick="changeContent()">Click Me!</button>
```

The Result in Browser

The script changes the heading text and makes it purple after the button is clicked. This simple interaction is the foundation of all dynamic web pages.

The JavaScript (`script.js`)

```
function changeContent() {
  // 1. Get the object using its ID
  const heading = document.getElementById('title');

  // 2. Change a property (the text content)
  heading.innerText = 'Content Changed by JS!';

  // 3. Change a style property (the color)
  heading.style.color = '#876cd4ff';
  heading.style.fontSize = '24px';
}
```

This process—**select** the element, then **manipulate** its properties or call its methods—is the fundamental loop of client-side web development.

Essential DOM Properties and Methods for Interaction

Once you select an element, you use these properties and methods to inject data, respond to users, and manage appearance.

innerText / innerHTML (Properties)

Used to get or set the text content of an element. `innerText` handles only text, while `innerHTML` can handle full HTML code (use with caution to avoid security issues).

classList (Property)

Provides methods (like `add()`, `remove()`, `toggle()`) to manipulate the CSS classes applied to an element, making it easy to change styles dynamically.

addEventListener() (Method)

The most critical method for interactivity. It waits for a specific event (like a 'click', 'mouseover', or 'keypress') on an element and then runs a specified function (the event handler).

appendChild() (Method)

Used to add new elements to the DOM. If you create a new `<li>` element, you use this method to attach it to its parent `<ul>` node.

Key Takeaways and Next Steps

You've taken the crucial step from static HTML/CSS to dynamic web development. Remember these key points as you continue your learning journey.

JavaScript: The Interactivity Layer

It's the language that controls the behavior of the webpage, allowing for real-time changes and user engagement.

The DOM: A Programmable Tree

The DOM turns HTML into a structured tree of objects that JavaScript can read and manipulate.

Objects, Properties, and Methods

This is the language of interaction: HTML elements are Objects, their attributes are Properties, and their actions are Methods.

Practice is Key

The best way to learn is by doing. Try creating a small project where clicking a button changes text, style, or adds a new element to the page.

> “Web development is about creating experiences. JavaScript and the DOM are the tools you need to build them.”