Document Object Model (DOM) in JavaScript

Unit-II

Web Technologies-II

Document Object Model (DOM)

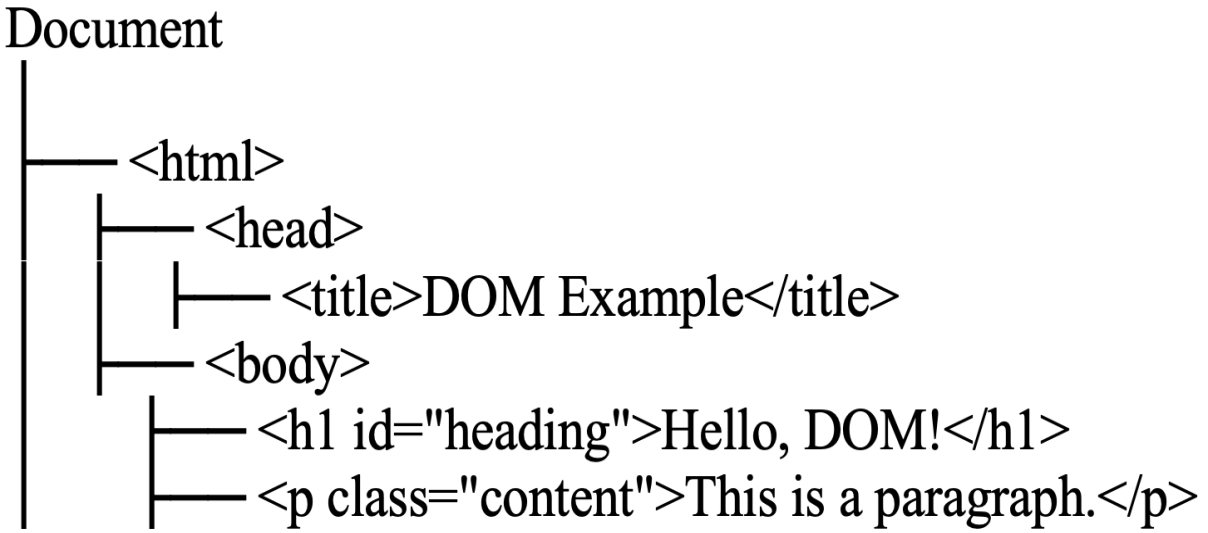
- The **Document Object Model (DOM)** is a programming interface for web documents.
- It represents the structure of an HTML or XML document as a tree of objects, allowing JavaScript to access and manipulate elements dynamically.
- The DOM is an **object-oriented representation** of a webpage that JavaScript can interact with.
- It allows **dynamic changes** to content, structure, and styles of a web page.

DOM Tree Structure

The DOM represents an HTML document as a **hierarchical tree structure**, where each element is a node.

```
<html><head>
<title>DOM Example</title>
</head>
<body>
<h1 id="heading">Hello, DOM!</h1>
<p class="content">This is a paragraph.</p>
</body></html>
```

Corresponding DOM Tree:



Accessing and Manipulating DOM Elements in JavaScript

The **Document Object Model (DOM)** is a structured representation of an HTML document that allows JavaScript to **access, manipulate, and modify** HTML elements dynamically.

- Accessing DOM Elements
- Manipulating DOM Elements

Accessing DOM Elements

a) Using document.getElementById()

•Selects an element using its id.
•Returns a **single** element.

Example:

```
let heading = document.getElementById("heading");
console.log(heading.innerHTML);

// Output: Hello, DOM!
```

Accessing DOM Elements

b) Using document.getElementsByClassName()

•Selects elements using their class name.
•Returns an **HTMLCollection** (array-like object).

Example:

```
let paragraphs = document.getElementsByClassName("content");
console.log(paragraphs[0].innerHTML);

// Output: This is a paragraph.
```

Accessing DOM Elements

c) Using document.getElementsByTagName()

•Selects elements using their tag name (e.g., h1, p).
•Returns an **HTMLCollection**.

Example:

```
let allParagraphs = document.getElementsByTagName("p");
console.log(allParagraphs.length); //
```

Output: Number of <p> elements

Accessing DOM Elements

d) Using document.querySelector()

•Selects **the first** matching element using **CSS selectors**.
•More flexible than getElementById().

Example:

```
let firstParagraph = document.querySelector("p");
console.log(firstParagraph.innerHTML);

// Output: This is a paragraph.
```

Accessing DOM Elements

e) Using document.querySelectorAll()

•Selects **all** matching elements as a **NodeList**.

Example:

```
let allParagraphs = document.querySelectorAll("p");
allParagraphs.forEach(p => console.log(p.innerHTML));

// Loops through all <p> elements
```

Manipulating DOM Elements

Once elements are accessed, JavaScript can modify their content, attributes, styles, and structure.

a) Changing HTML Content (innerHTML)
document.getElementById("heading").innerHTML = "Welcome to JavaScript!";

b) Changing Text Content (textContent)
•innerHTML parses HTML inside the element, while textContent only sets plain text.

document.getElementById("heading").textContent = "Updated Heading";

Manipulating DOM Elements

c) Changing Attributes (setAttribute() and getAttribute())

```
document.getElementById("heading").setAttribute("style", "color: red;");
console.log(document.getElementById("heading").getAttribute("style"));
// Output: color: red;
```

d) Modifying Styles (style property)

```
document.getElementById("heading").style.color = "blue";
document.getElementById("heading").style.fontSize = "24px";
```

e) Adding and Removing CSS Classes (classList)

```
document.getElementById("heading").classList.add("new-class");
// Add a class
document.getElementById("heading").classList.remove("old-class");
 // Remove a class
```

Creating and Removing Elements Dynamically

a) Creating a New Element

```
let newParagraph = document.createElement("p");  // Create a <p> element
newParagraph.textContent = "This is a dynamically added paragraph.";
document.body.appendChild(newParagraph); // Add to the document
```

b) Removing an Element

```
let heading = document.getElementById("heading");
heading.remove(); // Remove the <h1> element
```

Event Handling in DOM

JavaScript can respond to user interactions like clicks, keyboard input, and mouse movements.

a) Adding an Event Listener

```
document.getElementById("heading").addEventListener("click", function() {
    alert("Heading Clicked!");});
```

b) Changing Background Color on Button Click

```
<button onclick="changeColor()">Click Me</button>
<script>    function changeColor() {
      document.body.style.backgroundColor = "lightblue";    }
</script>
```

Traversing the DOM

DOM traversal refers to navigating the parent-child relationships between elements.

a) Accessing Parent Node

```
let childElement = document.getElementById("heading");
console.log(childElement.parentNode); // Returns the parent <body> element
```

b) Accessing Child Nodes

```
let parent = document.body;console.log(parent.children); // Returns all child elements
```

c) Accessing Sibling Elements

```
let firstParagraph = document.querySelector("p");
console.log(firstParagraph.nextElementSibling); // Next sibling
console.log(firstParagraph.previousElementSibling); // Previous sibling
```