JavaScript

Module 6 - Document Object Model (DOM)

Chapter 19 - Introduction to DOM

What is the Document Object Model (DOM)

Definition:

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of a document as a tree of objects, where each object corresponds to a part of the document, such as elements, attributes, and text.

What is the Document Object Model (DOM)

Core Concepts:

1.Document: The entire HTML or XML document is represented as the root of the DOM tree.

2.Element: HTML elements (e.g., ** **, **<p>**, **<h1>**) are represented as objects in the DOM tree.

3.Attribute: Attributes of HTML elements (e.g., **id**, **class**) are accessible as properties of DOM objects.

4.Node: Everything in the DOM is a node, including elements, attributes, and even text.

5.Tree Structure: The DOM represents the document as a hierarchical tree structure, where each node has a specific relationship with other nodes.

What is the Document Object Model (DOM)

Why DOM is Important:

- **Dynamic Web Pages**: The DOM enables dynamic manipulation of web pages using JavaScript. You can add, modify, or delete elements and attributes, creating interactive and responsive web applications.
- **Event Handling**: DOM events allow you to respond to user actions (clicks, keypresses, etc.) and update the document dynamically.
- **Data Exchange**: AJAX (Asynchronous JavaScript and XML) relies on the DOM to update parts of a web page without requiring a full page reload.

How the DOM represents web page structure

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sample Page</title>
  </head>
  <body>
    <h1>Hello, DOM!</h1>
    <p>This is a sample page.</p>
  </body>
</html>
```

- This HTML structure is represented as a DOM tree, with the **<html>** element as the root, containing child nodes like **<head>**, **<body>**, and so on.
- Understanding the DOM is essential for web developers to create dynamic and interactive web applications.

DOM Methods

Accessing Elements:

- **getElementById:** Retrieves an element by its unique ID.

```
const header = document.getElementById('header');
```

- **getElementsByClassName:** Retrieves elements by their class name.

```
const paragraphs = document.getElementsByClassName('paragraph');
```

- **getElementsByTagName:** Retrieves elements by their tag name.

```
const headings = document.getElementsByTagName('h1');
```

- **querySelector:** Retrieves the first element that matches a CSS selector.

```
const firstParagraph = document.querySelector('p');
```

- **querySelectorAll**: Retrieves all elements that match a CSS selector.

```
const allParagraphs = document.querySelectorAll('p');
```

Modifying Elements:

change the content, appearance, or behavior of existing elements on your web page after it has loaded.

innerHTML: Gets or sets the HTML content within an element.

```
header.innerHTML = 'New Header';
```

textContent: Gets or sets the text content within an element.

```
firstParagraph.textContent = 'Updated text';
```

setAttribute: Sets the value of an attribute for an element.

```
header.setAttribute('class', 'main-header');
```

Creating and Appending Elements:

createElement: Creates a new HTML element.

```
const newDiv = document.createElement('div');
```

appendChild: Appends a child element to another element.

```
document.body.appendChild(newDiv);
```

DOM Document

Document Object:

- The **document** object represents the entire HTML or XML document in the DOM.
- It serves as the entry point for accessing and manipulating the content of a web page.

Basic Properties:

- **document.title:** Gets or sets the title of the document.

```
const pageTitle = document.title;
document.title = 'New Title';
```

- **document.URL:** Gets the full URL of the document.

```
const pageURL = document.URL;
```

- **document.body:** Represents the **<body>** element of the document.

```
const bodyElement = document.body;
```

DOM Changing HTML Content

The easiest way to modify the content of an HTML element is by using the **innerHTML** property. To change the content of an HTML element, use this syntax:

document.getElementById(*id*).innerHTML = *new HTML*

```
<body>

    <h1 id="id01">Old Heading</h1>

    <!-- JavaScript block to change the content of the heading -->
    <script>
        // Find the element with the ID "id01"
        const element = document.getElementById("id01");

        // Change the inner HTML content of the element
        element.innerHTML = "New Heading";
    </script>

    <!-- You can include any other body elements here -->

</body>
```

- Inside the **<body>**, there is an **<h1>** element with the ID "id01" and the initial text "Old Heading".
- The **<script>** block contains JavaScript code that selects the element with the ID "id01" using **document.getElementById("id01")** and changes its inner HTML content to "New Heading" using **element.innerHTML = "New Heading";**.

DOM Forms

JavaScript Form Validation

- HTML form validation can be done by JavaScript.
- If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted.
- The function can be called when the form is submitted:

DOM Forms

```
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
```

```
<form name="myForm" action="/action_page.php" onsubmit="return validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

DOM Events

Events are actions or occurrences that happen in the browser, such as clicks, keypresses, or page load.

Event Handling

- The process of reacting to the events is called **Event Handling**.

```
const button = document.getElementById('myButton');

button.addEventListener('click', function() {
    // Your event handling logic here
});
```

DOM Events

- **Inline Event Handling:** Use inline event attributes in HTML .

```
<button onclick="handleClick()">Click me</button>
```

- **Using addEventListner():** used to attach an event handler to a particular element.

Click Event: Triggered when a mouse click occurs.

```
button.addEventListener('click', function() {
    // Click event handling logic
});
```

DOM Events

- **Submit Event**: Triggered when a form is submitted.

```
const form = document.getElementById('myForm');

form.addEventListener('submit', function(event) {
    event.preventDefault(); // Prevents the default form submission
    // Submit event handling logic
});
```

- **Keydown Event:**Triggered when a key is pressed down.

```
document.addEventListener('keydown', function(event) {
    // Keydown event handling logic
});
```

DOM Events

- **Mouseover Event**: Triggered when the mouse pointer enters an element.

```
const element = document.getElementById('myElement');

element.addEventListener('mouseover', function() {
    // Mouseover event handling logic
});
```

Thank You