Building Trust & Careers

DOM

DOCUMENT OBJECT MODEL

DOCUMENT OBJECT MODEL (DOM) MANIPULATION

Introduction to DOM:

What is the DOM?

- The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of the HTML document as a tree of nodes, where each node represents part of the document.
- The DOM allows programming languages (like JavaScript) to access, manipulate, and modify the structure, style, and content of web pages dynamically.

Why DOM Manipulation?

- DOM Manipulation is important for creating dynamic, interactive web pages. By manipulating the DOM, you can change the content, structure, and appearance of a webpage without needing to reload the page.

```
<div>
 <h1>Welcome to the DOM!</h1>
    <p>This is an example to demonstrate the DOM.</p>
    <button>Click Me!</button>
</div>
 <script>
      // Accessing the h1 element and changing its color using
JavaScript
      const heading = document.querySelector('h1');
      heading.style.color = 'blue'; // Change the color of the h1 tag
 </script>
```

Understanding DOM Structure:

The DOM represents the document as a tree structure, where:

- Elements (HTML tags) are represented as nodes.
- Each element has parent-child relationships.
- The root node represents the entire document (usually <html>).

```
<div id="parent">
    <h2>DOM Structure</h2>
    <p>This is a paragraph.</p>
    <button>Click Me!</button>
</div>

<script>
    // Accessing the parent div and its child h2 element
    const parentDiv = document.getElementById('parent');
    const heading = parentDiv.querySelector('h2');
    console.log(heading.textContent); // Logs: DOM Structure
</script>
```

Key Terms:

- Node: A single point in the DOM tree (can be an element, text, or attribute).
- Element: A type of node that represents an HTML tag (e.g., , <p>, etc.).
- Text Node: Represents the textual content inside an element.
- Attribute: A property of an element (like id, class, etc.)

Fetching Elements:

Getting Elements by ID:

- const element = document.getElementById('myElement');
- This returns the first element with the specified ID.
- Getting Elements by Class Name:

const elements

- const elements = document.getElementsByClassName('myClass');
- This returns a collection of elements with the specified class name.

Getting Elements by Tag Name:

- const elements = document.getElementsByTagName('div');
- This returns a collection of elements with the specified tag name.

Query Selector:

- const element = document.querySelector('.myClass');
- This returns the first matching element (CSS-style selector).

Query Selector All:

- const elements = document.querySelectorAll('.myClass');
- This returns a NodeList of all matching elements.

```
<ul>
    <li id="first">Apple</li>
    <li id="second">Banana</li>
    <li id="third">Cherry</li>
</ul>

<script>
    // Fetching a single element by ID
    const firstItem = document.getElementById('first');
    console.log(firstItem.textContent); // Logs: Apple

    // Fetching multiple elements by tag name
    const listItems = document.getElementsByTagName('li');
    console.log(listItems[1].textContent); // Logs: Banana
</script>
```

Manipulating DOM Elements:

Changing Content:

- Change text content:
- element.textContent = 'New Text';

Change HTML content:

- element.innerHTML = ' New Content ';

Changing Attributes:

- element.setAttribute('src', 'newImage.jpg');
- This changes the src attribute of an image element.

Changing Styles:

- element.style.color = 'blue';

```
<p id="demo">This text will change!</p>
<button id="changeTextButton">Change Text</button>

<script>
    // Selecting the paragraph and button
    const paragraph = document.getElementById('demo');
    const button = document.getElementById('changeTextButton');

    // Adding event listener to the button to change text when
clicked
    button.addEventListener('click', function() {
      paragraph.textContent = 'The text has changed!';
      paragraph.style.color = 'red'; // Change the text color to red
    });
</script>
```

Creating and Removing DOM Elements:

Creating New Elements:

- const newElement = document.createElement('div');
- newElement.textContent = 'Hello, world!';

Appending New Elements:

- document.body.appendChild(newElement);

Removing Elements:

- const elementToRemove = document.getElementById('elementToRemove');
- elementToRemove.remove();

Removing Child Elements:

- const parentElement = document.getElementById('parent');
- const childElement = parentElement.querySelector('.child');
- parentElement.removeChild(childElement);

```
<button id="createButton">Create New Element</button>
  <button id="removeButton">Remove Last Element</button>
  <ul id="list">
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
  <script>
    const list = document.getElementById('list');
    const createButton =
document.getElementById('createButton');
    const removeButton =
document.getElementById('removeButton');
    // Create a new list item when the button is clicked
    createButton.addEventListener('click', function() {
      const newItem = document.createElement('li'); // Create
new <li> element
      newItem.textContent = 'New Item'; // Set its content
      list.appendChild(newItem); // Add it to the list
    });

    // Remove the last list item when the button is clicked
    removeButton.addEventListener('click', function() {
      const lastItem = list.lastElementChild; // Get the last <li>
      if (lastItem) {
        list.removeChild(lastItem); // Remove it from the list
      }
    });
  </script>
```