# Day-7

## Task-1 : DOM Manipulation with JavaScript

The **Document Object Model (DOM)** represents the structure of an HTML or XML document as a tree of objects. Using JavaScript, we can interact with and manipulate the DOM to dynamically change the content, style, and behavior of a webpage.

### Key Concepts in DOM Manipulation

#### *1. Accessing DOM Elements*

To manipulate the DOM, you first need to access the elements in the document. JavaScript provides several methods to do this:

- **`getElementById`**: Selects an element by its `id`.

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

- **`getElementsByClassName`**: Selects all elements with a specific class (returns an HTMLCollection).

```
const buttons = document.getElementsByClassName('btn');
```

- **`getElementsByTagName`**: Selects all elements with a specific tag name.

```
const paragraphs = document.getElementsByTagName('p');
```

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

```
const firstButton = document.querySelector('.btn');
```

- **`querySelectorAll`**: Selects all elements that match a CSS selector (returns a NodeList).

```
const allButtons = document.querySelectorAll('.btn');
```

#### *2. Manipulating Content*

Once an element is selected, you can modify its content:

- **`innerHTML`**: Sets or gets the HTML content inside an element.

```
const div = document.getElementById('myDiv');
div.innerHTML = '<h1>Hello, World!</h1>';
```

- **`innerText`** or **`textContent`**: Sets or gets the text content of an element.

```
div.textContent = 'Hello, Text!';
```

### *3. Manipulating Attributes*

Attributes like `id`, `class`, `src`, etc., can be changed dynamically.

- **`setAttribute`**: Sets an attribute's value.

```
const link = document.querySelector('a');
link.setAttribute('href', 'https://www.example.com');
```

- **`getAttribute`**: Gets an attribute's value.

```
console.log(link.getAttribute('href'));
```

- **Direct property access**: Attributes can often be accessed as properties.

```
link.href = 'https://www.example.com';
```

### *4. Changing Styles*

You can modify an element's style using the `style` property.

- Directly set styles:

```
const button = document.querySelector('.btn');
button.style.backgroundColor = 'blue';
button.style.color = 'white';
```

- Add or remove classes:

```
button.classList.add('active');
button.classList.remove('inactive');
```

### *5. Adding or Removing Elements*

You can dynamically create or remove elements:

- **Create an Element**:

```
const newElement = document.createElement('p');
newElement.textContent = 'This is a new paragraph.';
document.body.appendChild(newElement);
```

- **Remove an Element**:

```
const elementToRemove = document.querySelector('p');
elementToRemove.remove();
```

### *6. Event Listeners*

Events allow you to interact with the DOM based on user actions.

- Add an event listener:

```
const button = document.querySelector('.btn');
button.addEventListener('click', () => {
    alert('Button clicked!');
});
```

- Remove an event listener:

```
function handleClick() {
    alert('Button clicked!');
}
button.addEventListener('click', handleClick);
button.removeEventListener('click', handleClick);
```

### *7. Traversing the DOM*

You can navigate the DOM tree to find related elements:

- **Parent Node**: `parentNode` or `parentElement`

```
const parent = element.parentNode;
```

- **Child Nodes**: `childNodes` or `children`

```
const children = element.children;
```

- **Siblings**: `nextSibling`, `previousSibling`

```
const next = element.nextElementSibling;
const previous = element.previousElementSibling;
```