**# JavaScript DOM Manipulation**

The **Document Object Model (DOM)** is a programming interface that allows JavaScript to interact with and modify HTML and CSS dynamically.

# Selecting Elements

## 1. getElementById()

Finds an element by its `id`.

```
const title = document.getElementById("main-title");
title.style.color = "blue"; // Changes the text color
```

## 2. getElementsByClassName()

Selects elements by class name (returns an HTMLCollection).

```
const items = document.getElementsByClassName("list-item");
console.log(items[0]); // Access first item
```

## 3. getElementsByTagName()

Selects elements by tag name (returns an HTMLCollection).

```
const paragraphs = document.getElementsByTagName("p");
console.log(paragraphs.length); // Total paragraph count
```

## 4. querySelector()

Selects the **first** matching element.

```
const header = document.querySelector("h1");
header.style.fontSize = "24px";
```

### 5. querySelectorAll()

Selects **all** matching elements (returns a NodeList).

```
const listItems = document.querySelectorAll(".list-item");
listItems.forEach(item => item.style.color = "red");
```

## Modifying Elements

### 1. innerHTML

Changes the **HTML content** of an element.

```
document.getElementById("content").innerHTML = "<h2>New Content</h2>";
```

### 2. textContent

Changes **text content** (ignores inner HTML).

```
document.getElementById("content").textContent = "New Text";
```

### 3. setAttribute() & getAttribute()

Sets or gets attributes of an element.

```
const link = document.querySelector("a");
link.setAttribute("href", "https://example.com");
console.log(link.getAttribute("href"));
```

### 4. classList.add(), remove(), toggle()

Manipulates CSS classes.

```
const box = document.querySelector(".box");
box.classList.add("highlight");
box.classList.remove("shadow");
box.classList.toggle("hidden");
```

# Handling Events

## 1. addEventListener()

Attaches an event listener.

```
const button = document.getElementById("myButton");
button.addEventListener("click", function() {
    alert("Button clicked!");
});
```

## 2. onClick Event

Inline JavaScript event.

```
<button onclick="alert('Button clicked!')">Click Me</button>
```

## 3. Mouse Events

```
button.addEventListener("mouseover", () => console.log("Mouse Over"));
button.addEventListener("mouseout", () => console.log("Mouse Out"));
```

## 4. Keyboard Events

```
document.addEventListener("keydown", event => {
    console.log("Key pressed: ", event.key);
});
```

# Creating and Removing Elements

## 1. createElement() & appendChild()

Creates and appends elements.

```
const newElement = document.createElement("p");
newElement.textContent = "New Paragraph";
document.body.appendChild(newElement);
```

## 2. removeChild()

Removes an element.

```
<div id="container">
  <p id="remove-me">This will be removed</p>
</div>

const parent = document.getElementById("container");
const child = document.getElementById("remove-me");
parent.removeChild(child);

Alternative Method (Modern Approach)
Instead of removeChild(), you can use element.remove():
document.getElementById("remove-me").remove();
```

# Summary

- **Selecting Elements**: Use `getElementById()`, `querySelector()`, etc.
- **Modifying Elements**: Change content using `innerHTML`, `textContent`, and attributes using `setAttribute()`.
- **Handling Events**: Use `addEventListener()` to attach events like clicks, mouse events, and keypresses.
- **Creating & Removing Elements**: Use `createElement()`, `appendChild()`, and `removeChild()`.

- The DOM is crucial for dynamic and interactive web applications.

By understanding DOM manipulation, developers can create dynamic, interactive web applications efficiently.