# Dom basics

<!DOCTYPE html>

<html>

<head>

<title>DOM Example</title>

<style>

.highlight {

background-color: yellow;

}

</style>

</head>

<body>

<h1 id="title">Hello, World!</h1>

<p class="text">This is a paragraph.</p>

<p class="text">This is another paragraph.</p>

<div class="container">

<p>Inside a div.</p>

</div>

<ul>

<li>List item 1</li>

<li>List item 2</li>

<li>List item 3</li>

</ul>

<button id="changeTextButton">Change Text</button>

<button id="addHighlightButton">Highlight Text</button>

<button id="createElementButton">Create Element</button>

<button id="removeElementButton">Remove Element</button>

<script>

// Accessing elements by ID

const title = document.getElementById('title');

console.log(title); // <h1 id="title">Hello, World!</h1>

// Accessing elements by class name

const paragraphs = document.getElementsByClassName('text');

console.log(paragraphs); // HTMLCollection of <p class="text"> elements

// Accessing elements by tag name

const listItems = document.getElementsByTagName('li');

console.log(listItems); // HTMLCollection of <li> elements

// Accessing elements using querySelector

const firstParagraph = document.querySelector('.text');

console.log(firstParagraph); // First <p class="text"> element

// Accessing elements using querySelectorAll

const allParagraphs = document.querySelectorAll('.text');

console.log(allParagraphs); // NodeList of <p class="text"> elements

// Changing innerHTML and textContent

const button = document.getElementById('changeTextButton');

button.addEventListener('click', () => {

title.innerHTML = 'Hello, DOM!';

});

// Setting and getting attributes

const addHighlightButton = document.getElementById('addHighlightButton');

addHighlightButton.addEventListener('click', () => {

firstParagraph.setAttribute('class', 'text highlight');

console.log(firstParagraph.getAttribute('class')); // text highlight

});

// Declare newDiv in the outer scope

let newDiv = null;

// Creating and appending elements

const createElementButton = document.getElementById('createElementButton');

createElementButton.addEventListener('click', () => {

newDiv = document.createElement('div');

newDiv.textContent = 'A new div element';

document.body.appendChild(newDiv);

});

// Removing elements

const removeElementButton = document.getElementById('removeElementButton');

removeElementButton.addEventListener('click', () => {

if (newDiv && document.body.contains(newDiv)) {

document.body.removeChild(newDiv);

newDiv = null; // Reset newDiv to prevent errors on subsequent clicks

} else {

console.log('No newDiv element to remove');

}

});

// Event handling

const container = document.querySelector('.container');

container.addEventListener('click', () => {

container.style.backgroundColor = 'lightblue';

});

</script>

</body>

</html>

## · Accessing Elements:

* document.getElementById(id): Returns the element with the specified ID.
* document.getElementsByClassName(class): Returns a collection of all elements in the document with the specified class name.
* document.getElementsByTagName(tag): Returns a collection of all elements in the document with the specified tag name.
* document.querySelector(selector): Returns the first element within the document that matches the specified group of selectors.
* document.querySelectorAll(selector): Returns a static NodeList of all elements in the document that match the specified group of selectors.

## · Manipulating Elements:

* element.innerHTML: Gets or sets the HTML content inside an element.
* element.textContent: Gets or sets the text content of an element.
* element.setAttribute(attr, value): Sets the value of an attribute on the specified element.
* element.getAttribute(attr): Gets the value of an attribute on the specified element.
* element.style.property: Gets or sets the value of a CSS property on the specified element.

## · Creating and Removing Elements:

* document.createElement(tagName): Creates an element with the specified tag name.
* element.appendChild(newNode): Adds a new child node to an element.
* element.removeChild(childNode): Removes a child node from an element.

## · Event Handling:

* element.addEventListener(event, function): Registers an event listener on the specified element.
* element.removeEventListener(event, function): Removes an event listener from the specified element.