# 📘 1. JavaScript DOM Selection

## 🔹 What is DOM?

* The **DOM (Document Object Model)** represents the structure of your HTML page as a tree.
* JavaScript uses the DOM to **select**, **read**, and **manipulate** elements on the page.

## 🔹 DOM Selection Methods Summary

| Method | Selector Type | Returns | Live? | Use Case |
| --- | --- | --- | --- | --- |
| getElementById() | id | Single element/null | ❌ No | When selecting **one unique element** |
| getElementsByClassName() | .class | HTMLCollection | ✅ Yes | To select **all elements** with a class |
| querySelector() | CSS selector | First match/null | ❌ No | For **one element** using any CSS selector |
| querySelectorAll() | CSS selector | NodeList (all) | ❌ No | To select **all matches** using CSS selectors |

## 🔹 1. getElementById("id")

* ✅ Selects **1 element** by ID
* ✅ Returns a **single element** or null
* <p id="title">Hello</p>
* let title = document.getElementById("title");
* title.style.color = "green";

## 🔹 2. getElementsByClassName("class")

* ✅ Selects **all elements** with a class
* ✅ Returns an **HTMLCollection** (live)
* <p class="note">Note 1</p>
* <p class="note">Note 2</p>
* let notes = document.getElementsByClassName("note");
* notes[1].style.color = "blue";

## 🔹 3. querySelector("selector")

* ✅ Selects the **first matching** element
* ✅ Works with #id, .class, tag, etc.
* ✅ Returns **single element** or null
* let title = document.querySelector("#title");
* let note = document.querySelector(".note");

## 🔹 4. querySelectorAll("selector")

* ✅ Selects **all elements** matching a CSS selector
* ✅ Returns a **NodeList** (not live)
* let items = document.querySelectorAll(".item");
* items[0].style.color = "red";
* ✅ Returns: **NodeList** (a list of all matching elements)
* ✅ Use: To get **multiple elements** using **CSS selectors**
* ✅ Not live (does **not auto-update** when DOM changes)

### 🧠 Example:

<p class="item">Item 1</p>

<p class="item">Item 2</p>

let items = document.querySelectorAll(".item");

console.log(items.length);     // 2

items[0].style.color = "red";  // Changes first item text to red

### ⚡ Quick Recap Table:

| Method | Selector Type | Returns | Live? |
| --- | --- | --- | --- |
| getElementsByClassName | Class name | HTMLCollection | ✅ Yes |
| getElementById | ID only | Single Element/null | ❌ No |
| querySelector | CSS selector | First Match/null | ❌ No |
| querySelectorAll | CSS selector | NodeList (all) | ❌ No |

# 📘 2. **innerText** vs **textContent** vs **innerHTML** in JavaScript

| Feature | innerText | textContent | innerHTML |
| --- | --- | --- | --- |
| 📋 Returns | Visible **text** only (no tags) | **All text**, incl. hidden | HTML + text |
| 👁️ Visibility | Respects CSS (display: none, etc.) | Ignores CSS | Shows raw HTML |
| 🧹 Whitespace | Cleaned up | Preserved | Preserved |

### 📌 Quick Examples:

    <h1 class="test">Hello this is lambodar <span style="display: none;">this is test text</span></h1>

test.innerText

'Hello this is lambodar'

test.textContent

'Hello this is lambodar this is test text'

test.innerHTML

'Hello this is lambodar <span style="display: none;">this is test text</span>'

### ✅ Best Use:

* Use innerText → for what's shown to users.
* Use textContent → for **pure text processing**.
* Use innerHTML → for **inserting/retrieving HTML**.

# 📘 3. getAttribute() vs setAttribute() in JavaScript

### getAttribute() vs setAttribute() in JavaScript

| Feature | getAttribute() | setAttribute() |
| --- | --- | --- |
| 📖 Purpose | Reads an attribute’s value | Sets/updates an attribute’s value |
| 🎯 Works On | Any HTML attribute (like href, id) | Any valid HTML attribute |
| 🧠 Use Case | Get custom or built-in attr values | Add/change attributes dynamically |
| ⚠️ Note | Doesn’t read **properties** like el.value | Doesn’t affect JS **properties** directly |

### 📌 Syntax:

element.getAttribute("attrName")

element.setAttribute("attrName", "value")

### ✅ Example:

<a id="link" href="https://example.com" target="\_blank">Click me</a>

const link = document.getElementById("link");

link.getAttribute("href");         // 🔎 "https://example.com"

link.setAttribute("href", "#");    // 🛠️ Changes href to "#"

link.setAttribute("title", "Go!"); // 🆕 Adds title attribute

### ❗ Common Attributes You Can Access:

* href, src, alt, id, class, type, name, value, title, data-\* attributes

## 🔧 Ways to Select Element + Use getAttribute() and setAttribute()

| Selector Method | Get Attribute Example | Set Attribute Example |
| --- | --- | --- |
| getElementById() | document.getElementById("myId").getAttribute("href") | document.getElementById("myId").setAttribute("href", "#") |
| getElementsByClassName() | document.getElementsByClassName("btn")[0].getAttribute("data-value") | document.getElementsByClassName("btn")[0].setAttribute("data-value", "123") |
| getElementsByTagName() | document.getElementsByTagName("img")[0].getAttribute("src") | document.getElementsByTagName("img")[0].setAttribute("alt", "Image") |
| querySelector() | document.querySelector(".card").getAttribute("title") | document.querySelector(".card").setAttribute("title", "Card Title") |
| querySelectorAll() | document.querySelectorAll("p")[0].getAttribute("id") | document.querySelectorAll("p")[0].setAttribute("id", "para1") |

### ✅ Reminder:

* .getAttribute("name") → gets the value
* .setAttribute("name", "value") → sets or updates the value

# 📘 4. How to Apply Styles in JavaScript

### ✅ 1. **Using .style.property**

element.style.color = "red";

element.style.backgroundColor = "yellow";

✔️ Directly applies inline styles

### ✅ 2. **Using .style.cssText** (set multiple at once)

element.style.cssText = "color: white; background: black; padding: 10px;";

### ✅ 3. **Using setAttribute("style", "...")**

element.setAttribute("style", "font-size: 18px; border: 1px solid gray;");

### ✅ 4. **Using classList.add() / remove() / toggle()**

*(Best for reusable styles via CSS classes)*

element.classList.add("active");

element.classList.remove("hidden");

element.classList.toggle("dark-mode");

### ✅ 5. **Using ES6+ Loop + Arrow Functions**

// Apply same style to multiple elements

document.querySelectorAll(".box").forEach(el => {

  el.style.borderRadius = "10px";

  el.style.boxShadow = "0 0 10px rgba(0,0,0,0.2)";

});

### 

### ✅ 6. **Using Object.assign()** (for bulk inline styles)

Object.assign(element.style, {

    color: "blue",

    fontWeight: "bold",

    marginTop: "20px"

  });

## ⚡ Most Used Style Properties in Real Projects:

* color, backgroundColor
* fontSize, fontWeight, textAlign
* margin, padding, border, borderRadius
* display, flex, grid
* width, height, position, zIndex

## 🎁 Bonus Tip — Toggle Dark Mode Example (ES6):

const toggle = document.getElementById("toggleTheme");

toggle.addEventListener("click", () => {

  document.body.classList.toggle("dark-mode");

});

# 📘 5. Access Parent Sibling & Children Elements

New trick to select element for every time we have to write id,class and then select element so easy way for testing your code go to chrome dev tool  
html code > hover on element >right click copy js path   
  
then just create var and pase this js path as selected element it will get selected and stores in var

### ✅ 1. **Parent Element**

* element.parentElement
* 🔄 Goes **one level up**
* ❗ Ignores text nodes (safe to use)

**Example:**

 <div class="cl">

        <h3>"hello world"</h3>

        <h3>"hello world"</h3>

        <h3>"hello world"</h3>

    </div>

----------------------------------------------------------------------------

const myel=document.querySelector('h3')

--myel

<h3>"hello world"</h3>

-- myel.parentElement

<div class="cl">…</div>

-- myel.parentElement.parentElement

<body>…</body>

-- myel.parentElement.parentElement.parentElement

<html lang="en"><head>…</head><body>…</body></html>

### ✅ 2. **Children Elements**

element.children

* Returns **HTMLCollection** of child elements (no text nodes)
* element.firstElementChild
* element.lastElementChild

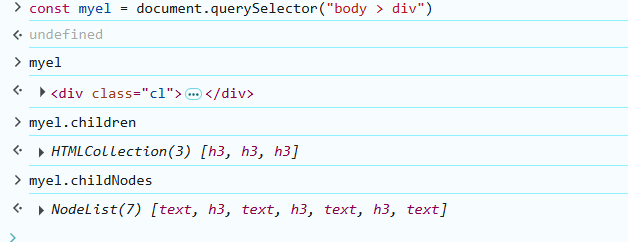
 <div class="cl">

        <h3>"hello world"</h3>

        <h3>"hello world"</h3>

        <h3>"hello world"</h3>

    </div>

----------------------------------------------------------------------------

### ✅ 3. **Sibling Elements**

element.previousElementSibling  
element.nextElementSibling

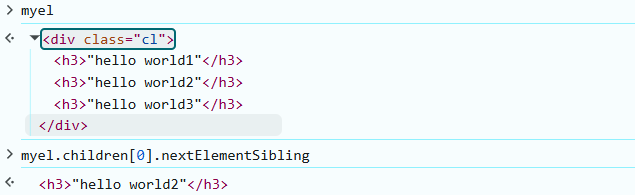
* Accesses elements **beside** the current one

**Example:**        <h3>"hello world"</h3>

        <h3>"hello world"</h3>

        <h3>"hello world"</h3>

    </div>



# 📘 6. Node vs Element — What's the Difference?

✅ Node ( General Building Block)

* A **generic** type — everything in the DOM is a Node.
* Types of nodes:
  + Element node (<div>, <p>)
  + Text node ("Hello")
  + Comment node (<!-- comment -->)
  + Document node (document)
  + etc.

**Example:**

document.body.childNodes;  
// Returns ALL nodes: elements, text, comments, etc.

✅ Element ( Specific Type of Node)

* A **node that is an HTML element**
* Subtype of Node
* Only represents tags like <div>, <p>, <ul>, etc.

**Example:**

document.body.children;  
// Returns ONLY element nodes (ignores text/comments)

🧠 Key Differences:

| Feature | Node | Element |
| --- | --- | --- |
| Type | General (includes text, etc.) | Specific (just HTML tags) |
| Example Nodes | Element, Text, Comment | <div>, <h1>, <ul>, etc. |
| Common Accessors | .childNodes, .firstChild | .children, .firstElementChild |
| Useful When | Parsing *everything* in DOM | Working with HTML *structure* |

✅ Quick Code Comparison:

// Includes text, comment, etc.  
console.log(document.body.childNodes);   
  
// Only actual HTML elements  
console.log(document.body.children);

# 📘 7. apendChild() & cloneNode() in JavaScript

### ✅ 1. appendChild()

Adds a node as the **last child** of a parent element.

parent.appendChild(child);

**Example:**

  <div class="cl1">

        <h3>"hello this is part of class1"</h3>

    </div>

    <div class="cl2">

        <h3>"hello this is part of class2"</h3>

    </div>

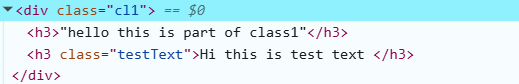
    <h3 class="testText">Hi this is test text </h3>

const class1= document.querySelector(".cl1")

const class2= document.querySelector(".cl2")

const testText= document.querySelector('.testText')

class1.appendChild(testText)



Example 2:

const list = document.querySelector("ul");

const newItem = document.createElement("li");

newItem.textContent = "New Item";

list.appendChild(newItem); // Adds <li> to end of <ul>

### 

### ✅ 2. element.cloneNode()

Creates a **copy of a node** (shallow or deep).

element.cloneNode(true);  // Deep clone (with children)

element.cloneNode(false); // Shallow clone (just the element)

**Example:**

const card = document.querySelector(".card");

const copy = card.cloneNode(true);

document.body.appendChild(copy); // Add the cloned card to body

### 📌 Common Use:

const node = document.getElementById("box");

const clone = node.cloneNode(true);  // Clone whole element

document.body.appendChild(clone);    // Insert at end of body

### 

### 🧠 Key Points:

| Method | Purpose | Notes |
| --- | --- | --- |
| appendChild() | Insert a node at the end | Moves existing or new node |
| cloneNode() | Clone element (optionally deep) | Doesn’t insert — needs append |

   <div class="container">

   <div class="card1 cardMain">c1</div>

   <div class="card2 cardMain">c2</div>

   <div class="card3 cardMain">c3</div>

   </div>

const selectCard= document.querySelector(".card1")

const container = document.querySelector(".container")

container.appendChild(selectCard.cloneNode(true))

<div class="card1 cardMain">c1</div>

