**DOM Manipulation & Dynamic Content Rendering**

**What is the DOM (Document Object Model)?**

When a web page loads, the browser turns the HTML into a tree-like structure. This structure is called the **DOM**.  
It helps JavaScript **see**, **read**, and **change** things on the web page.

**DOM Tree Structure:**

* **Nodes:** Everything in the DOM is a node (elements, text, attributes)
* **Elements:** HTML tags like <div>, <p>, etc.
* **Attributes:** Properties like id, class, src, etc.
* **Text:** Content inside HTML tags

**Selecting Elements in the DOM:**

**document.getElementById('id')**

Used to select an element by its ID.

let box = document.getElementById('box');

**document.querySelector('.class')**

Used to select the first element that matches a class, ID, or tag.

let button = document.querySelector('.btn');

**document.querySelectorAll('tag')**

Used to select **all** matching elements.

let allParagraphs = document.querySelectorAll('p');

**Modifying DOM Elements:**

**(Change Text or HTML Inside an Element)**

**textContent — Only changes the plain text inside an element**

<div id="box">Old text here</div>

let box = document.getElementById('box');

box.textContent = "Hello Students!";

✔ This changes the text inside the box, and removes any formatting.

**innerHTML — Changes the HTML content including formatting**

box.innerHTML = "<b>Hello</b> Students!";

✔ This allows you to use HTML tags like <b>, <i>, etc.

**Update Styles (like CSS)**

You can change styles directly using .style:

box.style.color = 'red'; // Changes text color to red

box.style.backgroundColor = 'yellow'; // Changes background color to yellow

✔ You can change other styles too like fontSize, margin, etc.

**Add or Remove Elements**

**Add a New Element**

let newDiv = document.createElement('div'); // Step 1: Create new div

newDiv.textContent = "This is new!"; // Step 2: Add some text

document.body.appendChild(newDiv); // Step 3: Add to the page

✔ A new div will be added at the bottom of the page with your text.

**Remove an Element**

<div id="removeMe">Remove this</div>

let child = document.getElementById('removeMe');

document.body.removeChild(child);

✔ This will remove the selected element from the page.

.

**JavaScript Events and Event Handling**

**1. What Are JavaScript Events?**

A **JavaScript event** is an action that happens in the browser, usually as a result of the user doing something.

**Examples of events:**

* A user **clicks** a button
* A user **submits** a form
* A user **presses a key**
* The mouse **moves over** an element
* The **page finishes loading**

These events allow you to make your website **interactive**. You can respond to events using **event handlers**.

**2. Common JavaScript Events**

| **Event Name** | **What It Does** |
| --- | --- |
| click | Happens when the user clicks an element |
| submit | Happens when a form is submitted |
| mouseover | Happens when the mouse hovers over something |
| keydown | Happens when the user presses a key |
| load | Happens when the page finishes loading |

**3. Event Handling Methods**

**Inline Event Handling**

You can write the event directly in the HTML tag.

<button onclick="sayHello()">Click Me</button>

<script>

function sayHello() {

alert("Hello from inline event!");

}

</script>

Not recommended for large projects.  
Good for simple tasks or beginners.

**Using addEventListener()**

This is the **modern and flexible** way to handle events in JavaScript.

<button id="myBtn">Click</button>

<script>

let btn = document.getElementById("myBtn");

btn.addEventListener("click", function () {

alert("Hello from addEventListener!");

});

</script>

This keeps JavaScript and HTML separate — which is cleaner and better practice.

**Removing an Event Listener Example**

<!DOCTYPE html>

<html>

<head>

<title>Remove Event Listener Example</title>

</head>

<body>

<button id="myBtn">Click Me</button>

<button id="removeBtn">Remove Click Event</button>

<script>

// Step 1: Get the button element

let btn = document.getElementById('myBtn');

let removeBtn = document.getElementById('removeBtn');

// Step 2: Define the event handler function

function showMsg() {

alert("Button was clicked!");

}

// Step 3: Add the click event listener to 'myBtn'

btn.addEventListener('click', showMsg);

// Step 4: When 'removeBtn' is clicked, remove the 'click' event listener from 'myBtn'

removeBtn.addEventListener('click', function() {

btn.removeEventListener('click', showMsg);

alert("Click event removed from the button.");

});

</script>

</body>

</html>

**Explanation:**

1. We have two buttons:
   * **"Click Me"** — which shows an alert when clicked.
   * **"Remove Click Event"** — which, when clicked, will stop the first button from showing the alert.
2. The function showMsg is the event handler for the first button’s click event.
3. We add the click event to the first button using addEventListener('click', showMsg).
4. When you click the **Remove Click Event** button, the event listener on the first button is removed using removeEventListener('click', showMsg). After this, clicking the first button does nothing.

**4. The Event Object**

When an event happens, JavaScript passes an **event object** to the function.  
This object contains information about **what happened**, **where**, and **how**.

**📌 Useful Properties of the Event Object:**

btn.addEventListener("click", function(event) {

console.log(event.type); // e.g. "click"

console.log(event.target); // Element that was clicked

console.log(event.currentTarget); // Element that listener is attached to

});

**5. Preventing Default Behavior**

Some elements (like forms or links) have default actions.

* **Example:** A form submits and reloads the page.
* To **stop** that from happening, use event.preventDefault().

**Example:**

<form id="loginForm">

<input type="text" placeholder="Enter name" />

<button type="submit">Login</button>

</form>

<script>

let form = document.getElementById("loginForm");

form.addEventListener("submit", function(event) {

event.preventDefault(); // Stop form from reloading the page

alert("Form submitted, but page didn't reload.");

});

</script>

**6. Stopping Event Propagation**

When you click on an element inside another element, the click event doesn't just stay on the clicked element. It also **moves up** to the parent elements. This is called **event propagation** or **event bubbling**.  
This is called **event bubbling** (events move up the DOM tree).

**Imagine this HTML:**

<div id="outer" style="background: lightgray; padding: 20px;">

<button id="inner">Click Me</button>

</div>

* You have a **button** inside a **div**.
* When you click the button, two things can happen:
  1. The **button** knows it was clicked.
  2. The **div** also reacts as if it was clicked (because the event bubbles up).

**Why Does This Happen?**

* The browser lets the event bubble up the DOM tree from the clicked element to its parents.
* So, if you have event handlers on both the button and the div, **both handlers will run** when you click the button.

**How to Stop This?**

You can stop this bubbling behavior by using:

event.stopPropagation();

This tells the browser:  
**“Do not let this event go up to parent elements.”**

**Complete Example with Explanation**

<div id="outer" style="background: lightgray; padding: 20px;">

<button id="inner">Click Me</button>

</div>

<script>

// Event listener on outer div

document.getElementById("outer").addEventListener("click", function () {

alert("Outer Div Clicked");

});

// Event listener on inner button

document.getElementById("inner").addEventListener("click", function (event) {

event.stopPropagation(); // Stop event from bubbling up

alert("Button Clicked");

});

</script>

**What happens here?**

* When you click the **button**:
  + The **button's event listener** runs and shows: "Button Clicked"
  + Because of event.stopPropagation(), the click event **does NOT** bubble up to the div.
  + So, the div’s alert ("Outer Div Clicked") **does not show**.

**Developing Dynamic User Interfaces**

**Creating Dynamic Elements:**

**What It Means:**You can create HTML elements (like <div>, <li>, etc.) using JavaScript after the page has loaded — without refreshing the page.

**Dynamically Adding Elements: document.createElement(), appendChild()**

<button onclick="addItem()">Add Item</button>

<ul id="itemList"></ul>

<script>

function addItem() {

let li = document.createElement("li"); // create a new list item element

li.textContent = "New Product"; // set the text inside the list item

document.getElementById("itemList").appendChild(li); // add the new item to the existing list

}

</script>

**Explanation:**  
This code creates a new <li> element dynamically each time the button is clicked. The createElement method makes the new element, and appendChild inserts it into the existing <ul> list. This way, the page updates without reloading.

**Manipulating Classes: classList.add(), classList.remove(), classList.toggle()**

**Why We Use It:**To change the style (like color, size, visibility) of an element without touching the CSS file.

<button onclick="changeColor()">Toggle Color</button>

<div id="box" class="square"></div>

<style>

.square { width: 100px; height: 100px; background: gray; }

.red { background: red; }

</style>

<script>

function changeColor() {

document.getElementById("box").classList.toggle("red");

}

</script>

**Explanation:**  
Here, clicking the button toggles the .red class on the box. This means if the box is gray, it changes to red, and if red, it changes back to gray. This is useful for dynamically changing styles without altering the CSS file directly.

**Interactive User Interfaces:**

**Show/Hide Elements: Using JavaScript to toggle visibility**

**Why We Use It:**To show or hide a message, section, or any element when needed.

<button onclick="toggleMessage()">Show/Hide</button>

<p id="msg" style="display: none;">Welcome to the site!</p>

<script>

function toggleMessage() {

let p = document.getElementById("msg");

if (p.style.display === "none") {

p.style.display = "block"; // show the message

} else {

p.style.display = "none"; // hide the message

}

}

</script>

// When button is clicked, this function checks if the paragraph is hidden.

// If it is hidden (display: none), it shows the paragraph (display: block).

// If it is visible, it hides the paragraph again by setting display to none.

**Explanation:**  
This example shows how to display or hide any HTML element by changing its display style property. It helps in creating interactive sections that appear only when needed.

**Modal Windows: Creating pop-ups with event-driven content**

**What It Is:**  
A small box that appears in front of the screen to show a message, form, or button.

<button onclick="openModal()">Open Modal</button>

<div id="modal" style="display:none; position:fixed; top:30%; left:30%; background:white; padding:20px; border:1px solid black;">

<p>This is a modal</p>

<button onclick="closeModal()">Close</button>

</div>

<script>

function openModal() {

document.getElementById("modal").style.display = "block"; // show modal

}

function closeModal() {

document.getElementById("modal").style.display = "none"; // hide modal

}

</script>

**Explanation:**  
A modal is a popup box that appears on top of the page content. This example opens and closes the modal by changing the CSS display property. Modals are commonly used for alerts, forms, or extra information without leaving the current page.

**Handling User Interactions:**

**Input Fields and Buttons: Managing user input, button clicks, form submissions**

**Purpose:**To take input from the user and show a result when a button is clicked.

<input type="text" id="nameInput" placeholder="Enter your name">

<button onclick="greet()">Greet</button>

<p id="output"></p>

<script>

function greet() {

let name = document.getElementById("nameInput").value; // get user input

document.getElementById("output").textContent = "Hello, " + name; // display greeting

}

</script>

**Explanation:**  
This simple example shows how to get user input from a text field and display a message when the button is clicked. It demonstrates the basics of handling input and updating the page dynamically.