DOM method

Topic / Method	Description	Basic Requirement	Syntax
document object	Represents the entire HTML document	HTML page loaded in browser	document.property / method **eg.** document.write("Welcome to DOM");
Element Selection Methods			
getElementById()	Finds **ONE** element using unique id	Element must have id attribute	document.getElementById("id") **eg.** document.getElementById("p1").innerHTML = "Hi";

getElementsByClassName()	Finds **ALL** elements with same class	Elements must share same class	document.getElementsByClassName("class")
			
			eg.
			var s = document.getElementsByClassName("c1");
			s[0].innerText = "Changed"; // To get first element
			
			or
			
			for (let i of s) {
			i.innerHTML = "Changed using for loop";
			} //For All elements
			
			Note: Use .length to find total elements used
			console.log(s.length)

querySelector()	Selects **FIRST** matching element	Valid CSS selector	document.querySelector("selector") **eg.** document.querySelector("p").style.color = "red"; For Element (“p”) ID: (“#a”) Class: (“ .b”) Combination : (div p, .box p)

querySelectorAll()	Selects **ALL** matching elements	Valid CSS selector	document.querySelectorAll("selector")
			
			eg.
			var s = document.querySelectorAll ("p");
			s[1].innerHTML = "Second"; // For one element on 1st Index
			or //For all Elements
			for (let i of s) {
			i.innerHTML = "Changed using for loop";
			}

Content Manipulation			
innerHTML	Gets/sets HTML content inside element	Valid HTML element	element.innerHTML = "html" **eg.** el.innerHTML = "Hello";
innerText	Gets/sets **only text** (no HTML)	Valid HTML element	element.innerText = "text" **eg.** el.innerText = "Hello";
Attribute Manipulation			
setAttribute()	Sets or changes attribute value	Element reference	element.setAttribute(name,value) **eg.** img.setAttribute("src","pic.jpg");
getAttribute()	Gets attribute value	Existing attribute	element.getAttribute(name) **eg.** p.getAttribute("title");
removeAttribute()	Removes attribute from element	Existing attribute	element.removeAttribute(name) **eg.** p.removeAttribute("title");
Style Manipulation			
Style Manipulation	Changes CSS styles dynamically	Element reference	element.style.property = value **eg.** p.style.backgroundColor = "yellow"; **Note:** Always use camelCase

EVENT HANDLING

Method: Function Call Without Parameter

Description: Calls a function without passing any argument

Requirement: Element ID must be known and fixed

Syntax/Example:

```
<p onclick="fun()" id="test">Hello</p>

<script>
function fun() {
   document.getElementById("test").innerHTML = "Mouse clicked";
}
</script>
```

Method: Function Call With this as Parameter

Description: Calls a function and passes the current element

Requirement: Element can be dynamic, reusable

Syntax/Example:

```
<script>
function fun2(x) {
   x.innerHTML = "Mouse Click";
}
</script>
<p onclick="fun2(this)">test</p>
```

Method: Inline JavaScript (No Function Call)

Description: JavaScript logic written directly inside HTML event

Requirement: Only small, simple logic

Syntax/Example:

```
<p onclick="this.style.color='red'; this.innerHTML='Mouse click';">Test</p>
```

Latest Approch

Method: addEventListener()

Description: used to **attach an event to an HTML element**.

Requirement: Event name and function as handler

Syntax/Example:

element.addEventListener("event", handler);

event → event type (e.g. "click", "mouseover")

handler → function reference used in addEventListener

Example: (function assignment to an event)

```
<button id="a"> Click Me</button>
<script>
const box = document.querySelector("#a");
const show = () => alert('Button Clicked');

box.addEventListener("click", show);
</script>
```

Note: Remove **"on"** keyword when using addEventListener() from traditional HTML event attributes.

For example: onclick HTML Attribute will called as "**click**" in addEventListener,

Similarly onmouseover called as mouseover, onfocus called as focus **etc**….

Mouse events

Event Name	Description	When to Use	Simple Example
onclick	Fires when mouse click is completed	Button clicks	<button **onclick**="alert('Clicked')">Click</button>
onmousedown	Fires when mouse button is pressed	Detect press	<div **onmousedown**="this.style.background='yellow'">Press
onmouseup	Fires when mouse button is released	Detect release	<div **onmouseup**="this.style.background='pink'">Release
onmouseover	Fires when mouse enters element	Hover effect	<h1 **onmouseover**="this.style.color='red'"> Hover</h1>
onmouseout	Fires when mouse leaves element	Remove hover	<h1 **onmouseout**="this.style.color='black'"> Out</h1>

Keyboard Events

Event Name	Description	When to Use	Simple Example
onkeydown	Fires when a user presses a key	Detect key press immediately	<input type="text" **onkeydown**="console.log('Key down')">
onkeypress	Fires when a user presses a key (deprecated in modern browsers)	Detect character input	<input type="text" **onkeypress**="console.log('Key pressed')">
onkeyup	Fires when a user releases a key	Detect final input or key release	<input type="text" **onkeyup**="console.log('Key up')">

Form events

Event Name	Description	When to Use	Simple Example
onfocus	Fires when a field gets focus	Highlight or validate field on focus	<input type="text" **onfocus**="this.style.background='lightyellow'">
oninput	Fires immediately when field value changes	Live validation or dynamic feedback	<input type="text" **oninput**="this.value=this.value.toUpperCase()">
onblur	Fires when a field loses focus	Validate input after leaving field	<input type="text" **onblur**="alert('Left field')">
onchange	Fires when value changes and field loses focus	Detect final value change	<input type="text" **onchange**="alert('Value changed')">
onsubmit	Fires when a form is submitted	Validate form before submission	<form **onsubmit**="alert('Form submitted')"> <input type= "text"…..> <input type= "submit"> </form>

Methods to access values of Form elements

```
<body>
<form name="f1" onsubmit="return fun()">
   <input type="text" name="t1" id="i1" />
   <input type="submit"/>
</form>
<script>
function fun(){
    // METHOD1
    var obj = document.f1.t1.value;
    // METHOD2
   // var obj = document.forms["f1"]["t1"].value;
   // METHOD3
   // var obj = document.getElementById("i1").value;
   alert(obj);
}
</script>
</body>
```

Note: document.getElementById().value is the most reliable and recommended approach in modern JavaScript.

Event Object & Handling Essentials

Property	Description	Used For	Example
event.target	Element that triggered event	Identify element	<button onclick="show(event)"> Click</button> function show(e){ alert(**e.target.tagName**); } **Note:** Can fetch value also for <input> by using e.target.value
event.type	Type of event	Debugging	Change above to **alert(e.type);**

event.key	Key pressed	Keyboard input	<body
			onkeypress="keyC(event)">
			<script>
			function keyC(e) {
			alert(e.key);
			}
			</script>
			</body>

event.keyCode	Keycode of key	Keyboard input	Change above to **alert(e.keyCode);**
event.button	Value of Button click 0 : Left button 1 : Wheel or middle button (if present) 2 : Right button	Mouse click	<body onmousedown="fun(event)"> <script type="text/javascript"> function fun(e){ alert(e.button) } </script> </body>

event.preventDefault()	Stops default behavior	Form control	<form onsubmit="stop(event)"
			action='a.png'>
			<input type="submit"
			value="Submit">
			</form>
			<script>
			function stop(e) {
			e.preventDefault();
			// Will prevent form to redirect
			alert("Form submission stopped");
			}</script>