Element-Only Navigation

These properties help you move between **elements only** (ignores text nodes, comments).

a) parentElement

Returns the parent **element**.

```
element.parentElement
```

b) children

Returns all **element children** (not text nodes).

```
let list = document.querySelector("ul");
console.log(list.children);
```

c) firstElementChild

Returns the first **element** child.

```
element.firstElementChild
```

d) lastElementChild

Returns the last **element** child.

```
element.lastElementChild
```

e) nextElementSibling

Returns the next **element sibling**.

```
element.nextElementSibling
```

f) previousElementSibling

Returns the previous **element sibling**.

```
element.previousElementSibling
```

Table Navigation (Important for HTML tables)

Used for `<table>` elements.

a) table.rows

Returns all table rows (`<tr>`).

```
document.querySelector("table").rows;
```

b) table.tBodies

Returns all `<tbody>` sections.

c) row.cells

Returns all `<td>` or `<th>` inside a row.

```
let firstRow = table.rows[0];
console.log(firstRow.cells);
```

d) table.caption, table.tHead, table.tFoot

Access specific table sections.

Links Collection (Anchor Elements)

document.links

Returns all `<a>` tags with `href`.

```
console.log(document.links);
```

Searching Methods (Most Important)

(A) `document.getElementById()`

- Returns the element with that **ID**
- Fastest searching method
- ID must be unique

```
let title = document.getElementById("mainTitle");
```

(B) document.getElementsByClassName()

- Returns a live HTMLCollection
- Multiple items possible

```
let items = document.getElementsByClassName("item");
```

(C) document.getElementsByTagName()

- Returns elements by tag name

```
let paragraphs = document.getElementsByTagName("p");
```

(D) document.querySelector()

- Returns the **first matching element**

- Supports full CSS selectors

```
document.querySelector(".box");
document.querySelector("#title");
document.querySelector("ul li:nth-child(2)");
```

(E) document.querySelectorAll()

- Returns **all matching** elements (NodeList)

```
document.querySelectorAll("p");
```

Useful Element Methods: matches(), closest(), contains()

These are used in real-world DOM operations.

1) element.matches(selector)

Checks **if the element itself** matches the given CSS selector.

Example:

```
let btn = document.querySelector("button");

if (btn.matches(".primary")) {
    console.log("This button is primary");
}
```

Meaning:

- Returns **true** if the element fits the selector
- Very useful in event delegation

2) element.closest(selector)

Finds the **nearest ancestor** (parent → grandparent → up) that matches the selector.

Example:

```
let item = document.querySelector("li");

let list = item.closest("ul");
console.log(list); // nearest UL parent
```

Meaning:

- Searches upward
- Returns **first matching ancestor**
- If no match → returns `null`

3) element.contains(childNode)

Checks if an element **contains** another element.

Example:

```
let box = document.querySelector("#box");
let p = document.querySelector("#box p");

console.log(box.contains(p));  // true
```

Meaning:

- Returns **true** if child is inside parent
- Works for **descendants at any level**

6. Short Summary Table

Method	Meaning	Returns
getElementById	Find by ID	Single element
querySelector	First match	Single element
querySelectorAll	All matches	NodeList
matches()	Element matches selector?	true/false
closest()	Nearest ancestor matching selector	Element/null
contains()	Parent contains child?	true/false
parentElement	Parent element	Element
children	Only element children	HTMLCollection
nextElementSibling	Next element	Element
firstElementChild	First element child	Element

DOM Events (Basics)

Events are actions performed by the user or browser. Common events:

Event	Meaning
`click`	User clicks
`dblclick`	Double-click
`input`	User types in input
`change`	Input value changed
`mouseover`	Mouse on element
`mouseout`	Mouse leaves element
`keydown`	Key pressed
`submit`	Form submitted
`load`	Page loaded

Add event listener:

```
element.addEventListener("click", function () {
    console.log("Clicked!");
});
```

console.dir()

Used to inspect an element as a **JavaScript object**.

```
console.dir(document.body);
```

Shows:

- properties
- methods
- events
- attributes

Used more than `console.log()` for DOM debugging.

tagName and nodeName

tagName

- Works only for **element nodes**
- Always returns **UPPERCASE tag name**

```
let p = document.querySelector("p");
console.log(p.tagName); // "P"
```

nodeName

- Works for **all node types**
- Element node → tag name
- Text node → "#text"
- Comment node → "#comment"

```
console.log(p.firstChild.nodeName); // "#text"
```

innerHTML

Returns or sets the **HTML inside** an element.

```
div.innerHTML = "<b>Hello</b>";
```

Includes tags + text.

outerHTML

Returns or replaces the **entire element including itself**.

```
div.outerHTML = "<p>Replaced Div</p>";
```

textContent

Returns only **plain text**, no HTML tags.

```
div.textContent = "Hello World";
```

hidden Property

- A boolean that hides an element when set to `true`.

```
element.hidden = true;   // hides element
element.hidden = false;  // shows element
```

Same as adding `hidden` attribute in HTML.

Attributes and attribute methods

Attributes are key-value pairs in HTML.

Example:

```
<input id="box" type="text">
```

a) getAttribute(name)

```
box.getAttribute("type");
```

b) setAttribute(name, value)

```
box.setAttribute("placeholder", "Enter name");
```

c) removeAttribute(name)

```
box.removeAttribute("type");
```

d) hasAttribute(name)

```
box.hasAttribute("id"); // true
```

Data Attributes (dataset)

Used to store custom data inside HTML.

Example:

```
<div id="product" data-id="101" data-name="Laptop"></div>
```

Access in JS:

```
let p = document.getElementById("product");

console.log(p.dataset.id);
```

```
console.log(p.dataset.name);
```

Set value:

```
p.dataset.price = "50000";
```

DOM Insertion Methods

1) append()

Adds element **at the end** inside parent.

```
parent.append(child);
```

2) prepend()

Adds element **at the beginning** inside parent.

```
parent.prepend(child);
```

3) before()

Adds element **before** the current element.

```
element.before(newNode);
```

4) after()

Adds element **after** the current element.

```
element.after(newNode);
```

5) replaceWith()

Replaces an element.

```
element.replaceWith(newNode);
```

insertAdjacentHTML / Element / Text

These allow inserting content in 4 positions:

Position	Meaning
`"beforebegin"`	Before the element
`"afterbegin"`	Inside, at the start
`"beforeend"`	Inside, at the end
`"afterend"`	After the element

Example:

```
element.insertAdjacentHTML("beforebegin", "<p>Before</p>");
element.insertAdjacentHTML("afterbegin", "<p>Start</p>");
element.insertAdjacentHTML("beforeend", "<p>End</p>");
element.insertAdjacentHTML("afterend", "<p>After</p>");
```

Node Removal Methods

1) remove()

Remove element itself.

```
element.remove();
```

2) removeChild()

Remove a specific child.

```
parent.removeChild(child);
```

className

Gets or sets the **full class string**.

```
div.className = "box highlight";
```

Replaces all classes.

classList Methods

Most important, used in real world.

a) add()

```
div.classList.add("active");
```

b) remove()

```
div.classList.remove("active");
```

c) toggle()

Adds class if missing, removes if present.

```
div.classList.toggle("active");
```

d) contains()

Checks if class exists.

```
div.classList.contains("active"); // true/false
```

Quick Summary (Super Useful for Revision)

- `console.dir()` → inspect element object
- `tagName` / `nodeName` → type of element/node
- `innerHTML` → HTML inside
- `outerHTML` → full element
- `textContent` → text only
- `hidden` → hide/show
- Attributes → get/set/remove/has
- dataset → custom data attributes
- insertions → append, prepend, before, after, replaceWith
- insertAdjacentHTML → 4 insertion positions
- remove() / removeChild() → delete nodes
- className → full string
- classList → add, remove, toggle, contains

Browser Events

Events represent actions performed in the browser.

Common Event Types

Event	Meaning
`clic`	Mouse click
`dblclick`	Double click
`mouseover`	Mouse enters element
`mouseout`	Mouse leaves element
`keydown`	Key is pressed
`keyup`	Key released
`input`	User typing in input field
`submit`	Form submission
`scroll`	User scrolls page
`load`	Page has loaded
`contextmenu`	Right-click

Event Handling in JavaScript

A. addEventListener()

Modern way to attach an event.

Syntax

```
element.addEventListener(event, handler, options);
```

Example

```
document.getElementById("btn").addEventListener("click", () => {
  alert("Button clicked!");
});
```

Advantages

- Can attach multiple events.
- Cleaner and modern.
- Supports options like `once`, `capture`.

Example with options

```
btn.addEventListener("click", hello, { once: true });
```

Runs only **one time**.

B. removeEventListener()

Used to remove event handlers.

Important

To remove, handler **must be a named function**, not anonymous.

Example

```
function greet() {
  console.log("Hello");
}

btn.addEventListener("click", greet);
btn.removeEventListener("click", greet);
```

The Event Object

Whenever an event occurs, the browser sends an **event object** to the handler.

Example

```
document.addEventListener("click", function(event) {
  console.log(event);
});
```

Common Properties

Property	Meaning
`event.type`	Event name (click, keydown)

Property	Meaning
`event.target`	Element where event happened
`event.currentTarget`	Element whose handler is running
`event.key`	Which key was pressed (keyboard)
`event.clientX / clientY`	Mouse X, Y position
`event.preventDefault()`	Stop default behavior
`event.stopPropagation()`	Stop event bubbling

Inline vs Modern Event Handling

Inline (Old way)

```
<button onclick="hello()">Click</button>
```

Modern way (Recommended)

```
button.addEventListener("click", hello);
```

Event Flow

Events move in 3 phases:

1. **Capturing Phase** (top → target)
2. **Target Phase**
3. **Bubbling Phase** (target → top)

Example to stop bubbling

```
event.stopPropagation();
```

Keyboard Event Example

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

Form Submit Event

```
form.addEventListener("submit", (e) => {
  e.preventDefault();   // stops form from reloading
  console.log("Form Submitted");
});
```

10. Mouse Position Example

```
document.addEventListener("mousemove", (e) => {
  console.log(e.clientX, e.clientY);
});
```