Complete JS Course Syllabus

🔥 JavaScript – Learn Everything Series

📘 Phase 2: DOM, Events, Forms & Storage

🌐 Chapter 9: The DOM (Document Object Model)

🧠 What is the DOM?

The **DOM (Document Object Model)** is a programming interface that represents an HTML document as a **tree structure**.
JavaScript uses the DOM to **access, modify, add, and remove elements dynamically**.

The DOM converts HTML into **objects that JavaScript can understand and control**.

Think of the DOM as the **brain of the webpage** — JavaScript talks to the DOM, and the DOM updates the UI.

🌳 DOM Tree Structure

Every HTML document is converted into a **tree of nodes**:

- **Node** → Base unit of DOM
- **Element Node** → HTML tags (`div`, `p`, `button`)
- **Text Node** → Text inside elements
- **Comment Node** → HTML comments

Example:

```
document
 └── html
      ├── head
      └── body
           └── h1
                └── "Hello World"
```

🔍 Selecting DOM Elements

By ID

```
document.getElementById("title")
```

By Class Name

```
document.getElementsByClassName("item")
```

➡ Returns **HTMLCollection (live)**

By CSS Selector

```
document.querySelector(".box")
document.querySelectorAll(".box")
```

➡ `querySelectorAll` returns **NodeList (static)**

📝 Text & Content Properties

innerText

- Returns **only visible text**
- Affected by CSS

textContent

- Returns **all text**, including hidden
- Faster and more reliable

innerHTML

- Returns **HTML + text**
- Can inject tags

⚠️ Avoid `innerHTML` with user input → security risk

🧬 Attribute Manipulation

```
element.getAttribute("src")
element.setAttribute("id", "main")
element.removeAttribute("disabled")
```

Used to control **element behavior**, not styling.

🔧 Dynamic DOM Manipulation

```
document.createElement("li")
parent.appendChild(child)
parent.prepend(child)
parent.removeChild(child)
```

Used in:

- Todo apps
- Dynamic lists
- Interactive UI updates

🎨 Styling Elements

Inline Styles

```
element.style.backgroundColor = "red"
```

classList (Best Practice)

```
element.classList.add("active")
element.classList.remove("active")
element.classList.toggle("active")
```

⚠️ Common Confusions

innerText vs textContent vs innerHTML

Property	Hidden Text	HTML

innerText	❌	❌
textContent	✅	❌
innerHTML	✅	✅

HTMLCollection vs NodeList

- **HTMLCollection** → Live, auto-updates
- **NodeList** → Static snapshot

🧠 Mindset

DOM is **expensive to manipulate**.
Make **minimal changes**, batch updates, and avoid unnecessary reflows.

🧪 Practice Zone (DOM)

1. Create a list dynamically from an array
2. Add a new list item on button click
3. Remove a specific element using DOM methods
4. Change class on button click using `classList.toggle()`
5. Show difference between `innerText` and `textContent`
6. Select all buttons and change their text
7. Create and delete elements dynamically

🖱️ Chapter 10: Events & Event Handling

🧠 What are Events?

Events are **actions triggered by the user or browser**, such as clicking, typing, or submitting a form.

JavaScript listens to events and executes code in response.

🔗 Event Binding

```
element.addEventListener("click", handler)
element.removeEventListener("click", handler)
```

🖱️ Common Events

- `click`
- `input`
- `change`
- `submit`
- `mouseover`
- `keyup`

🎯 Event Object

Every event provides an object with useful information:

```
event.target
event.type
event.preventDefault()
```

🌊 Event Bubbling & Capturing

- **Bubbling (default)** → child → parent

- **Capturing** → parent → child

```
addEventListener("click", fn, true)
```

🧠 Event Delegation

Attach **one listener to a parent**, detect child using `event.target`.

✔ Better performance
✔ Works with dynamic elements

⚠️ Common Confusions

- `event.target` → element clicked
- `event.currentTarget` → element with listener
- Bubbling happens by default
- Capturing is rarely used

🧠 Mindset

Don't attach listeners to every element.
Delegate smartly.

🧪 Practice Zone (Events)

1. Create a button click counter
2. Show mouse position on `mousemove`
3. Implement event bubbling example
4. Use event delegation for list items
5. Prevent default form submission

6. Log `target` vs `currentTarget`
7. Create a live character counter

📝 Chapter 11: Forms & Form Validation

🧠 Forms in JavaScript

Forms collect user data.
JavaScript validates and sanitizes this data before submission.

📥 Reading Input Values

```
input.value
textarea.value
select.value
```

🚫 Prevent Default Submission

```
form.addEventListener("submit", e => {
  e.preventDefault()
})
```

✅ Validation Techniques

HTML Validation

- `required`
- `pattern`
- `minlength`

JavaScript Validation

- Regex checks
- Custom error messages

🚨 Error Handling

- Show error text dynamically
- Highlight invalid fields
- Remove error when fixed

⚠️ Common Confusions

- `value` → form elements
- `textContent` → normal elements
- Submit ≠ button click

🧠 Mindset

Forms are **communication bridges**.
Validate deeply, never trust user input.

🧪 Practice Zone (Forms)

1. Email validation using regex
2. Password strength checker
3. Show error messages dynamically
4. Disable submit until form is valid
5. Highlight invalid fields

6. Create feedback form with validation

🕐 Chapter 12: Timers & Intervals

⏳ setTimeout

Runs code **once after delay**

```
setTimeout(fn, 3000)
clearTimeout(id)
```

🔁 setInterval

Runs code **repeatedly**

```
setInterval(fn, 1000)
clearInterval(id)
```

⚠️ Common Confusion

- `setInterval` may drift
- Recursive `setTimeout` is more accurate

🧠 Mindset

Time controls **UX rhythm** — loaders, alerts, animations.

🧪 Practice Zone (Timers)

1. Countdown timer
2. Auto-hide alert after 3 seconds

3. Digital clock
4. Stop interval on button click
5. Loader that disappears after delay

📦 Chapter 13: Storage (LocalStorage, SessionStorage & Cookies)

🗄️ localStorage

- Persistent storage
- Data survives reload

```
localStorage.setItem("key","value")
localStorage.getItem("key")
localStorage.removeItem("key")
localStorage.clear()
```

🕘 sessionStorage

- Tab-based storage
- Cleared on tab close

📦 JSON Handling

Storage supports **only strings**:

```
JSON.stringify(obj)
JSON.parse(str)
```

🍪 Cookies (Basics)

- Stored as key=value
- Sent with every request
- Needs manual expiration

⚠️ Common Confusions

- localStorage is NOT secure
- Cookies are small & limited

🧠 Mindset

Use:

- localStorage → UI state
- cookies → authentication

🧪 Practice Zone (Storage)

1. Save theme preference
2. Remember username after reload
3. Store object using JSON
4. Clear storage on logout
5. Dark mode toggle using localStorage