JavaScript & the DOM

Web Technologies — Lecture 6

Masoud Hamad

State University of Zanzibar (SUZA)

Semester II, 2025/2026

Outline

JavaScript: The Language of the Web

- Created by Brendan Eich in **10 days** (1995) at Netscape
- Now standardised as **ECMAScript** (latest: ES2024)
- Runs in every browser AND on the server (Node.js, Deno, Bun)
- The *only* language native to the browser

In this course

We'll write **modern JS** (let/const, arrow functions, async/await, modules) — not legacy ES5.

Variables, Types, Functions

```
const PI = 3.14159;          // immutable binding
let count = 0;               // mutable
// var: avoid (function-scoped, hoisted)

// Types: number, string, boolean, null, undefined,
//        object, symbol, bigint
const name = "Amina";
const ok = true;
const arr = [1, 2, 3];
const obj = { id: 1, name: "Ali" };

// Function declaration
function add(a, b) { return a + b; }

// Arrow function (no own `this`)
const square = x => x * x;
```

Control Flow & Truthy/Falsy

```
if (count > 0) {
  console.log("positive");
} else if (count === 0) {        // === : strict equality
  console.log("zero");
} else {
  console.log("negative");
}

// Falsy: false, 0, "", null, undefined, NaN
// Everything else is truthy

const name = userInput || "Guest";    // default
const len = list?.length ?? 0;        // optional + nullish
```

Arrays & Objects

```
const nums = [1, 2, 3, 4];
nums.push(5);                          // [1,2,3,4,5]
const doubled = nums.map(n => n * 2);
const evens = nums.filter(n => n % 2 === 0);
const sum = nums.reduce((a, b) => a + b, 0);

// Destructuring
const [first, ...rest] = nums;
const { name, email } = user;

// Spread / rest
const merged = { ...user, role: "admin" };
```

What is the DOM?

DOM = Document Object Model.

- Browser parses HTML into a **tree of nodes**
- JS can read and modify this tree — and the browser re-renders
- Every HTML element becomes a JavaScript object

The DOM tree

```
document
  → <html>
    → <head> ...
    → <body>
      → <h1>, <p>, <div>, ...
```

Selecting Elements

```
// By id
const title = document.getElementById("title");

// CSS selectors (most flexible)
const btn   = document.querySelector(".save-btn");
const all   = document.querySelectorAll("ul li");

// Modern: NodeList works with for...of
for (const li of all) {
  console.log(li.textContent);
}
```

Reading & Modifying Elements

```
// Read
const text = el.textContent;
const html = el.innerHTML;
const value = input.value;

// Write
el.textContent = "New text";          // safer than innerHTML
input.value = "";

// Attributes
img.setAttribute("alt", "logo");
const href = link.getAttribute("href");

// Classes
el.classList.add("active");
el.classList.remove("hidden");
el.classList.toggle("open");

// Inline style
el.style.color = "red";
```

Creating & Inserting Nodes

```
const li = document.createElement("li");
li.textContent = "New item";
li.classList.add("todo-item");

document.querySelector("ul").appendChild(li);

// Or, more readable for many children:
ul.insertAdjacentHTML("beforeend", `
  <li class="todo-item">${escape(name)}</li>
`);

// Remove
li.remove();
```

Event Listeners

```
const btn = document.querySelector("#submit");

btn.addEventListener("click", (event) => {
  event.preventDefault();              // stop default behaviour
  console.log("Clicked!", event.target);
});

// Common events:
// click, input, change, submit, keydown, keyup,
// mouseenter, mouseleave, scroll, load, DOMContentLoaded
```

Event Delegation

Instead of attaching a listener to every list item, attach **one** to the parent.

```
document.querySelector("ul").addEventListener("click", (e) => {
  const li = e.target.closest("li");
  if (!li) return;
  console.log("Clicked item:", li.dataset.id);
});
```

Why: works for items added *later*, fewer listeners, better performance.

Callbacks → Promises → async/await

```
// Promise
fetch("/api/users")
  .then(res => res.json())
  .then(users => render(users))
  .catch(err => console.error(err));

// async/await -- reads top to bottom
async function loadUsers() {
  try {
    const res = await fetch("/api/users");
    const users = await res.json();
    render(users);
  } catch (err) {
    console.error(err);
  }
}
```

ES Modules

```
// math.js
export function add(a, b) { return a + b; }
export const PI = 3.14159;

// app.js
import { add, PI } from "./math.js";
console.log(add(2, 3));
```

In HTML:

```
<script type="module" src="app.js"></script>
```

A Tiny Todo App

```
const input = document.querySelector("#new");
const list  = document.querySelector("#todos");

document.querySelector("#form").addEventListener("submit", (e) => {
  e.preventDefault();
  const text = input.value.trim();
  if (!text) return;

  const li = document.createElement("li");
  li.textContent = text;
  li.addEventListener("click", () => li.classList.toggle("done"));
  list.appendChild(li);

  input.value = "";
  input.focus();
});
```

Summary

- JavaScript = the only language native to the browser
- The **DOM** represents your HTML as a tree of objects
- Use `querySelector`/`querySelectorAll` + event listeners
- Prefer event delegation for dynamic lists
- Use `async/await` for asynchronous work (network, files)
- ES Modules (`import`/`export`) for code organisation

Next: AJAX, Fetch & consuming REST APIs.

Questions?