**Seminarul 4 — Partea 1 (Teorie): Module, evenimente și DOM — ES modules & dependency management**

# Hook realist: „Lansăm luni dimineața”

Vineri seara, asociația studențească primește confirmarea că luni la ora 08:00... (versiunea completă este livrată în textul din chat).

Răspunsul nu este doar „import/export”... (versiunea completă este livrată în textul din chat).

# 1. De ce ES modules (ESM) în browser?

ES modules oferă izolare lexicală... (detaliile complete sunt în textul din chat).

// src/math/add.js  
export function add(a, b) { return a + b; }

// src/app.js  
import { add } from './math/add.js';  
console.log(add(2, 3)); // 5

// counter.js  
export let count = 0;  
export function inc(){ count++; }

// use-counter.js  
import { count, inc } from './counter.js';  
console.log(count); // 0  
inc();  
console.log(count); // 1

## 1.1. Exporturi implicite vs nominale

// user/formatName.js  
export default function formatName(u) { return `${u.last}, ${u.first}`; }

// or, named:  
export function formatName(u) { return `${u.last}, ${u.first}`; }

## 1.2. Re‑exports (agregare de module)

// services/index.js  
export { loadEvents } from './data.js';  
export { topInterests } from './report.js';

## 1.3. Import dinamic (`import()`) și top‑level await

// lazy-search.js  
export async function search(q){  
 const { buildIndex } = await import('./indexer.js');  
 const idx = await buildIndex();  
 return idx.find(q);  
}

// tla-example.mjs  
const data = await fetch('/data.json').then(r => r.json());  
export const initial = data;

# 2. Graful de module și rezoluția importurilor

<!-- index.html -->  
<script type="importmap">  
{  
 "imports": {  
 "lodash-es": "https://cdn.skypack.dev/lodash-es"  
 }  
}  
</script>  
<script type="module">  
 import { uniq } from "lodash-es";  
 console.log(uniq([1,1,2]));  
</script>

# 4. DOM: arhitectură, querying, actualizare eficientă

// src/dom.js  
export const qs = (sel, root=document) => {  
 const el = root.querySelector(sel);  
 if (!el) throw new Error(`Missing element: ${sel}`);  
 return el;  
};  
export const qsa = (sel, root=document) => [...root.querySelectorAll(sel)];

// src/renderList.js  
export function renderList(container, items){  
 const frag = document.createDocumentFragment();  
 for (const it of items){  
 const li = document.createElement('li');  
 li.textContent = it.label;  
 frag.appendChild(li);  
 }  
 container.innerHTML = '';  
 container.appendChild(frag);  
}

# 5. Modelul de evenimente: delegare, opțiuni, AbortController

// src/delegate.js  
export function on(container, type, selector, handler, options){  
 const listener = (ev) => {  
 const t = ev.target.closest(selector);  
 if (!t || !container.contains(t)) return;  
 handler(ev, t);  
 };  
 container.addEventListener(type, listener, options);  
 return () => container.removeEventListener(type, listener, options);  
}

// src/cleanup.js  
export function listenWithSignal(el, type, cb, signal, options){  
 el.addEventListener(type, cb, { ...options, signal });  
}

# 6. Securitate & integritate: CSP

<meta http-equiv="Content-Security-Policy"  
 content="default-src 'self'; script-src 'self' 'strict-dynamic' 'nonce-XYZ'; object-src 'none'; base-uri 'none'">

# 7. Performanță & UX: modulepreload, throttle

<link rel="modulepreload" href="/src/main.js">

// throttling simplu  
export const throttle = (fn, ms=50) => {  
 let last = 0;  
 return (...args) => {  
 const now = Date.now();  
 if (now - last >= ms) { last = now; fn(...args); }  
 };  
};

# 8. Testabilitate: Vitest & Jest (jsdom)

// tests/vitest/dom.test.js  
import { describe, it, expect, beforeEach } from 'vitest';  
import { renderList } from '../../src/renderList.js';  
describe('renderList', () => {  
 let ul;  
 beforeEach(() => { document.body.innerHTML = '<ul id="list"></ul>'; ul = document.querySelector('#list'); });  
 it('populates items', () => {  
 renderList(ul, [{label:'A'},{label:'B'}]);  
 expect(ul.querySelectorAll('li').length).toBe(2);  
 });  
});

// tests/jest/dom.jest.test.js  
import { describe, it, expect, beforeEach } from '@jest/globals';  
import { renderList } from '../../src/renderList.js';  
describe('renderList (Jest)', () => {  
 let ul;  
 beforeEach(() => { document.body.innerHTML = '<ul id="list"></ul>'; ul = document.querySelector('#list'); });  
 it('populates items', () => {  
 renderList(ul, [{label:'A'},{label:'B'}]);  
 expect(ul.querySelectorAll('li').length).toBe(2);  
 });  
});

# Bibliografie (APA 7, cu DOI)

Wirfs‑Brock, A., & Eich, B. (2020). JavaScript: The first 20 years. Proceedings of the ACM on Programming Languages, 4(HOPL), 1–189. https://doi.org/10.1145/3386327

Loring, M. C., et al. (2017). Semantics of asynchronous JavaScript. ACM SIGPLAN Notices, 52(11), 51–62. https://doi.org/10.1145/3170472.3133846

Maffeis, S., Mitchell, J. C., & Taly, A. (2008). An operational semantics for JavaScript. In Programming Languages and Systems (pp. 307–325). Springer. https://doi.org/10.1007/978-3-540-89330-1\_22

Ungar, D., & Smith, R. B. (1991). SELF: The power of simplicity. Higher‑Order and Symbolic Computation, 4(3), 171–216. https://doi.org/10.1007/BF01806105