**Seminarul 5 — CSS Selectors & Layout  
Partea 1: Teorie (extins)**

**Hook realist: "Hub-ul Asociațiilor Studențești" în 15 minute, fără framework-uri**

Imaginați-vă că, la ora 08:30, primiți un mesaj de la coordonatorul asociațiilor studențești:   
„La 10:00 avem un stand deschis pentru noii studenți; pune, te rog, online o pagină simplă cu   
carduri pentru cluburi, un formular de înscriere și un layout 3×3 care să arate bine pe orice ecran.”   
Nu aveți voie să folosiți framework‑uri CSS (Bootstrap/Tailwind). Aveți doar \*\*CSS selectors\*\* bine alese,   
pseudo‑clase moderne (de ex. `:focus-visible`, `:has()`), \*\*form styling\*\* accesibil și un \*\*grid 3×3\*\*  
care colapsează elegant pe mobil. În 90 de minute, veți livra o pagină curată, lizibilă și predictibilă.   
Acesta este exercițiul mental al seminarului: să formăm un mod de gândire orientat pe \*\*selectors\*\*,   
\*\*stări\*\* și \*\*pattern‑uri\*\* de layout — astfel încât nevoile reale să devină simple compoziții CSS.

**1. Cascadă, Specificitate, Moștenire: modelul mental al CSS**

CSS este un limbaj de \*\*rezolvare a conflictelor\*\*: mai multe reguli pot viza același element, iar   
\*\*cascada\*\* decide care regulă câștigă; \*\*specificitatea\*\* (ID > clasă/atribut/pseudo‑clasă > tip/pseudo‑element)   
determină prioritatea între regulile aflate la același „nivel” în cascadă; \*\*moștenirea\*\* asigură transmiterea   
unor proprietăți (de exemplu `color`, `font-family`) către descendenți. În practică, obiectivul nostru este să   
obținem stiluri \*\*predictibile\*\* cu \*\*specificitate mică\*\*, pentru a facilita extensia și întreținerea.   
Evităm `!important` (cu excepții bine motivate) și preferăm compoziții clare de clase și selectori.

Trucuri utile de disciplină:  
• mențineți \*\*lanțurile de selectori\*\* puțin adânci (1–2 combinatori),  
• preferați \*\*clase de utilitate\*\* pentru ajustări frecvente,  
• folosiți `:where()` ca „neutralizator de specificitate” în prefața lanțurilor.

:root {  
 --space-2: 0.5rem; --space-4: 1rem; --ring: 2px solid #1a73e8;  
}  
/\* Straturi de specificitate redusă \*/  
:where(.btn, .tag, .chip) { border-radius: .5rem; }  
  
/\* Claritate în combinare (sibling + child combinators) \*/  
.card { display: grid; gap: var(--space-4); }  
.card > h3 + p { margin-top: calc(var(--space-2) \* -1); }

**2. Taxonomia selec­torilor (CSS Selectors) — de la fundament la `:has()`**

\*\*De bază\*\*: selectori de tip (element), de clasă (`.class`) și de ID (`#id`), în combinație cu   
\*\*combinatori\*\*: `A B` (descendent), `A > B` (copil), `A + B` (adjacent), `A ~ B` (frați generali).   
\*\*Selectori de atribute\*\*: `[attr]`, `[attr="value"]`, `[attr^="pre"]`, `[attr$="suf"]`, `[attr\*="sub"]`,   
și semnalizatorul \*\*case‑insensitive\*\* `i` (`[attr="value" i]`).

\*\*Pseudo‑clase structurale\*\*: `:first-child`, `:last-child`, `:only-child`, `:nth-child(an+b)`,   
`:nth-of-type()`, `:empty`, `:not()`.   
\*\*Pseudo‑clase de interacțiune/focus\*\*: `:hover`, `:active`, `:focus`, `:focus-visible`, `:focus-within`, `:target`.   
\*\*Pseudo‑clase UI/Form\*\*: `:checked`, `:disabled/:enabled`, `:required/:optional`, `:valid/:invalid`,   
`:in-range/:out-of-range`, `:placeholder-shown`, `:read-only/:read-write`.   
\*\*Pseudo‑clase moderne pentru compoziție\*\*: `:is()` și `:where()` (diferență cheie: \*\*specificitatea\*\* –   
`:where()` contribuie cu 0).   
\*\*`:has()`\*\* (relațional): permite „stilarea părintelui” în funcție de existența unui descendent potrivit.

/\* Grupare cu :is() / :where() pentru scăderea specificității \*/  
:where(button, .btn, [role="button"]):is(:hover, :focus-visible) {  
 outline: var(--ring);  
 outline-offset: 2px;  
}  
  
/\* Parent styling cu :has(): grup de formular devine 'invalid' dacă are input invalid \*/  
.form-group:has(input:invalid),  
.form-group:has(select:invalid) {  
 --border: #b00020;  
 border-color: var(--border);  
 background: color-mix(in oklab, var(--border) 6%, white);  
}

**3. Form styling (UI states) — accesibil, predictibil, coerent**

Form‑urile sunt locul în care \*\*state‑urile UI\*\* contează: `:placeholder-shown`, `:required/:optional`,   
`:valid/:invalid`, `:focus/:focus-visible`, `:disabled`, `:read-only` etc. Un formular robust are \*\*focus ring\*\*   
vizibil, \*\*spațiere\*\* suficientă, \*\*contrast\*\* adecvat (WCAG recomandă minim 4.5:1 pentru text normal),   
și \*\*mesaje de eroare\*\* lângă câmpurile problematice. Plasarea mesajelor în proximitatea câmpului reduce timpul   
de corecție și crește eficiența, lucru susținut de studii controlate cu \*eye tracking\* asupra formularelor   
și asupra complexității vizuale (de ex., Hall & Hanna, 2004; Tuch et al., 2009; Seckler et al., 2014).

/\* Focus ring coerent și diferit de :hover \*/  
:where(input, select, textarea, button):focus-visible {  
 outline: var(--ring);  
 outline-offset: 2px;  
}  
  
/\* Stări de validare nativă: \*/  
input:required:invalid { border-color: #b00020; }  
input:required:valid { border-color: #1e8e3e; }  
  
/\* Parent highlight cu :has() \*/  
.field:has(input:focus) { box-shadow: 0 0 0 2px color-mix(in oklab, #1a73e8 20%, transparent); }  
  
/\* Accenting nativ pentru checkbox/radio \*/  
input[type="checkbox"], input[type="radio"] { accent-color: #1a73e8; }  
  
/\* File input \*/  
input[type="file"]::file-selector-button {  
 padding: .5rem .75rem;  
 border: 1px solid #ccc;  
 background: #f9f9f9;  
 cursor: pointer;  
}

\*\*Contrastul\*\* rămâne un factor cognitiv major pentru lizibilitate și reținere: studii clasice arată impactul   
combinațiilor culoare‑text asupra citirii și memoriei, iar literatura HCI recomandă limite minime de contrast   
(WCAG) superioare standardelor ISO pentru a acoperi variabilitatea percepțiilor (de exemplu, deficiențe de culoare).

**4. Layouturi moderne cu CSS Grid — cazul 3×3**

\*\*Grid\*\* este soluția naturală pentru layouturi bidimensionale (rânduri × coloane). Un \*\*grid 3×3\*\*   
clasic pentru carduri de cluburi poate fi definit cu `repeat(3, 1fr)` și `gap`, apoi făcut \*\*responsiv\*\* cu `auto-fit`   
și `minmax()` sau cu \*media queries\*, astfel încât să colapseze la 2 coloane și apoi la 1 pe ecrane înguste.   
Pentru raportare coerentă de imagini, folosiți `aspect-ratio`. Pentru mapări semantice ale zonelor, utilizați   
`grid-template-areas`.

.cards {  
 --gap: 1rem;  
 display: grid;  
 grid-template-columns: repeat(3, 1fr);  
 gap: var(--gap);  
}  
@media (max-width: 900px) {  
 .cards { grid-template-columns: repeat(2, 1fr); }  
}  
@media (max-width: 520px) {  
 .cards { grid-template-columns: 1fr; }  
}  
  
/\* Variante cu auto-fit + minmax \*/  
.cards-auto {  
 display: grid;  
 grid-template-columns: repeat(auto-fit, minmax(16rem, 1fr));  
 gap: 1rem;  
}  
.card\_\_media { aspect-ratio: 4 / 3; object-fit: cover; }

\*\*Grid areas\*\* clarifică intenția și documentează layoutul în sine (auto‑documentare).

.layout {  
 display: grid;  
 grid-template-areas:  
 "hd hd hd"  
 "sb main main"  
 "ft ft ft";  
 grid-template-columns: 16rem 1fr 1fr;  
 grid-template-rows: auto 1fr auto;  
 gap: 1rem;  
}  
.header { grid-area: hd; } .sidebar { grid-area: sb; }  
.main { grid-area: main; } .footer { grid-area: ft; }

**5. Accesibilitate & ergonomie cognitivă (legătura cu HCI)**

Două principii HCI oferă ancore utile pentru decizii CSS orientate pe \*\*focus\*\* și \*\*densitate\*\*:   
(1) \*\*Fitts’ law\*\* — timpul de atingere a unei ținte crește cu distanța și scade cu dimensiunea (mai exact,   
depinde de raportul dintre distanță și lățimea țintei); (2) \*\*Hick–Hyman law\*\* — timpul deciziei crește   
logaritmic cu numărul de alternative. În practică: \*\*asigurați ținte suficient de mari\*\* (buton/form control) și   
\*\*evitați supraîncărcarea vizuală\*\* în grid‑uri sau meniuri. Cercetări despre \*\*complexitatea vizuală\*\* confirmă   
asocieri între aglomerare, scăderea eficienței și memorie mai slabă; CSS oferă remedii concrete (spațiere,   
ierarhii vizuale, culori cu contrast bun, separatoare blânde).

Pentru contrastul perceput, literatura de culoare propune metrice precum \*\*CIEDE2000\*\*; în web, algoritmul   
WCAG de contrast (bazat pe luminanță relativă sRGB) rămâne standard practic, dar înțelegerea metricei CIEDE2000   
ajută la raționamente cromatice în temele (dark/light) și în diferențierea badge‑urilor.

**6. Performanță & întreținere — de ce ne pasă de simplitatea selec­torilor**

CSS poate declanșa recalculări de \*\*style\*\*, \*\*layout\*\* (reflow) și \*\*paint\*\*/compoziție. Deși stilurile sunt,   
în general, ieftine, \*\*lanțurile de selectori foarte adânci\*\* și pattern‑urile care forțează schimbări frecvente de   
layout pot penaliza. Practic: mențineți selectori \*\*simpli și puțini\*\*, evitați animarea proprietăților care declanșează   
layout (de exemplu `width`, `height`), preferând `transform`/`opacity`, și folosiți `contain` acolo unde izolați sub‑arbori   
DOM („containers”).

article { contain: content; } /\* limitează efectele recalculelor în interiorul articolului \*/

**7. BEM, utilitare și tokenizare (CSS custom properties)**

Pentru proiecte didactice și reale, \*\*BEM\*\* oferă o gramatică de numire predictibilă. \*\*CSS custom properties\*\*   
— ca „design tokens” (`--color-primary`, `--space-4`, `--radius-sm`) — permit tematici consistente (light/dark) și   
ajută la reducerea specificității prin concentrarea deciziilor în root și în blocuri.

.btn { --bg: #1a73e8; --fg: white; background: var(--bg); color: var(--fg); }  
.btn--secondary { --bg: #5f6368; }

**8. Exemple aplicate pe scenariu (Clubs & Associations Hub)**

\*\*Carduri în grid 3×3\*\* cu stări `:hover/:focus-visible` bine definite; \*\*formular de înscriere\*\* cu `:has()`   
pentru evidențierea grupurilor cu erori; \*\*badge‑uri\*\* pentru statut (de ex., „înscrieri deschise”) randate cu `::before`;   
\*\*link‑uri de salt\*\* (`#details`) și `:target` pentru a sublinia secțiunea activă. Fiecare micro‑decizie folosește   
selectors cu specificitate minimă, ușor de extins.

/\* Grid 3x3 + card states \*/  
.cards { display: grid; gap: 1rem; grid-template-columns: repeat(3, 1fr); }  
.card { background: white; border: 1px solid #e5e7eb; border-radius: .75rem; padding: 1rem; }  
.card:where(:hover, :focus-within) { border-color: #1a73e8; box-shadow: 0 2px 8px rgba(0,0,0,.06); }  
.card\_\_badge::before { content: "Open"; display: inline-block; padding: .125rem .5rem; background: #1e8e3e; color: white; border-radius: 999px; font-size: .75rem; }

**9. AI‑assist (Copilot/LLM) — promturi VSL de lucru real**

Exemple de promturi scurte, iterative (\*\*VSL\*\* = \*very short loop\*) pentru a accelera munca fără a pierde controlul:  
  
• „Generează un set minimal de selectori pentru focus vizibil pe linkuri, butoane și inputuri, \*\*folosind\*\* `:where()` pentru specificitate 0 și `outline-offset: 2px`.”  
  
• „Propune un \*\*grid 3×3\*\* responsiv (3→2→1) cu `minmax(16rem, 1fr)` și exemplu de card cu `aspect-ratio: 4/3`.”  
  
• „Transformă acest lanț de selectori adânci într‑o variantă echivalentă cu \*\*maxim doi combinatori\*\* și clase semantic‑clare (`.card`, `.field`, `.form-group`).”  
  
• „Scrie un set de reguli `:has(input:invalid)` pentru a evidenția \*\*doar\*\* grupurile cu erori și o zonă `.errors[role="status"]` pentru sumar.”  
  
• „Sugerează \*\*tokens\*\* (`--space-\*`, `--radius-\*`, `--color-\*`) pentru theming light/dark și arată cum schimb `--color-primary` în funcție de `prefers-color-scheme`.”

**10. Checklist teoretic (ce ar trebui să stăpâniți după această parte)**

• Puteți explica \*\*cascade/specificitate/moștenire\*\* și să justificați evitarea `!important`.  
  
• Cunoașteți \*\*selectori moderni\*\*: `:is()`, `:where()`, `:has()`, diferențele lor de specificitate și cazurile de utilizare.  
  
• Știți să proiectați \*\*form styling\*\* robust (focus ring, stări nativ‑valide/invalid, accent‑color, `::file-selector-button`).  
  
• Puteți compune un \*\*grid 3×3\*\* responsiv, cu \*grid areas\* și \*aspect-ratio\* pentru media.  
  
• Înțelegeți legătura cu \*\*HCI\*\* (Fitts, Hick, complexitate vizuală) și \*\*contrastul\*\* (WCAG, CIEDE2000 ca fundal teoretic).  
  
• Aveți reflexe de \*\*performanță\*\* (contain, evitarea animării layout‑ului) și de \*\*întreținere\*\* (BEM, tokens).

**Referințe (APA, cu DOI)**

Hall, R. H., & Hanna, P. (2004). The impact of web page text‑background colour combinations on readability, retention, aesthetics, and behavioural intention. \*Behaviour & Information Technology, 23\*(3), 183–195. https://doi.org/10.1080/01449290410001669932

Tuch, A. N., Bargas‑Avila, J. A., Opwis, K., & Wilhelm, F. H. (2009). Visual complexity of websites: Effects on users’ experience, physiology, performance, and memory. \*International Journal of Human‑Computer Studies, 67\*(9), 703–715. https://doi.org/10.1016/j.ijhcs.2009.04.002

Seckler, M., Heinz, S., Bargas‑Avila, J. A., Opwis, K., & Tuch, A. N. (2014). Designing usable web forms: Empirical evaluation of web form improvement guidelines. \*Proceedings of the 2014 CHI Conference on Human Factors in Computing Systems\*. https://doi.org/10.1145/2556288.2557265

Hick, W. E. (1952). On the rate of gain of information. \*Quarterly Journal of Experimental Psychology, 4\*(1), 11–26. https://doi.org/10.1080/17470215208416600

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. \*Journal of Experimental Psychology, 47\*(6), 381–391. https://doi.org/10.1037/h0055392

Liu, W., Zeng, F., & Chen, X. (2022). Eye tracking, usability, and user experience: A systematic literature review. \*International Journal of Human‑Computer Interaction, 38\*(15), 1478–1505. https://doi.org/10.1080/10447318.2021.2002044

Sharma, G., Wu, W., & Dalal, E. N. (2005). The CIEDE2000 color‑difference formula: Implementation notes, supplementary test data, and mathematical observations. \*Color Research & Application, 30\*(1), 21–30. https://doi.org/10.1002/col.20070