**Seminarul 6 — CSS Layout & Media Queries  
Partea 1: Teorie (extins)**

**Hook realist: patru ecrane, un singur CSS (de la telefon la panou 4K)**

La ora 08:00 primești o cerință: până la prânz trebuie să publici o pagină de prezentare pentru „Clubs & Associations Hub”. Va rula pe patru tipuri de ecrane aflate în campus: telefoane mici (320–400 px), tablete (≈768–1024 px), laptopuri de laborator (≈1280 px) și un panou 4K din hol. Nu ai voie framework‑uri; trebuie să livrezi doar cu \*\*CSS\*\*: \*\*Flexbox\*\*, \*\*Grid\*\*, \*\*media queries\*\*, imagini nedistorsionate, spațiere coerentă și tipografie lizibilă. Nu există timp pentru „încercări norocoase” („pixel pushing”), ci doar pentru decizii bune, reproductibile.  
În acest seminar învățăm să gândim layout‑ul „de jos în sus” (\*\*mobile‑first\*\*), cu reguli stabile, \*\*unități fluide\*\* și \*\*funcții CSS\*\* moderne, să alegem \*\*breakpoints\*\* în funcție de conținut (nu de „device marketing”) și să combinăm \*\*Flexbox\*\* cu \*\*Grid\*\* în mod sistematic. Vom lega deciziile de principii HCI (ergonomie, viteză de scanare vizuală) și vom include „micro‑tooling” mental: checklist‑uri, convenții de denumire și teste automate care confirmă practicile implementate.

**1. Responsiv, dar riguroaş: mobile‑first, content‑out, ritm vizual**

„Responsive” nu înseamnă „mai multe media queries”. Înseamnă \*\*conținut care dictează layout‑ul\*\*. Abordarea \*\*mobile‑first\*\* începe cu un flux vertical curat, apoi \*\*relaxează constrângerile\*\* pe măsură ce spațiul crește. Din punct de vedere cognitiv, citirea eficientă pe ecrane mici presupune o \*\*lățime de text\*\* rezonabilă (de regulă 60–80 de caractere pentru paragrafe), \*\*spațiere verticală\*\* suficientă pentru a delimita blocurile, și un \*\*contrast\*\* confortabil între text și fundal. Aceste decizii sunt tehnic susținute de unități fluide și funcții CSS moderne, care reduc numărul de praguri (breakpoints) necesare.

Unitățile fluide („se dilată” odată cu containerul/viewportul) și funcțiile ajutătoare:  
• `%` – proporțional cu dimensiunea containerului; util pentru lățimi și spațieri dependente de context.  
• `vw`/`vh` – fracții din viewport (1 vw = 1% din lățimea viewport‑ului); folosite cu prudență la tipografie.  
• `fr` – unitatea „fraction” din Grid: împarte spațiul rămas pe coloane/rânduri.  
• `min()`/`max()`/`clamp()` – funcții pentru „garduri” elastice: setăm limite minime și maxime, reducând nevoia de media queries.  
• `ch` – aproximativ lățimea caracterului „0”; util pentru a controla lățimea măsurii de text (reading measure).

/\*\*\*\*\* Tipografie fluidă (exemplu) \*\*\*\*\*/  
:root{  
 --step-0: clamp(1rem, 0.9rem + 0.5vw, 1.125rem); /\* text de corp \*/  
 --step-2: clamp(1.25rem, 1.1rem + 1vw, 1.75rem); /\* subtitluri \*/  
 --step-4: clamp(1.6rem, 1.2rem + 2.2vw, 2.6rem); /\* titluri \*/  
}  
body{ font-size: var(--step-0); line-height: 1.6; }  
h2{ font-size: var(--step-2); }  
h1{ font-size: var(--step-4); max-width: 35ch; } /\* control al măsurii textului \*/

Prin folosirea `clamp()` pe scara tipografică și pe spațieri („spacing scale”), obținem \*\*fluiditate între praguri\*\*, astfel încât media queries să devină doar corecții locale („tuning”), nu cârje omniprezente. Acest lucru reduce complexitatea CSS și riscul de inconsistențe între vizualizări.

**2. Flexbox în profunzime: axa principală, ambalare (wrap) și aliniere**

\*\*Flexbox\*\* excelează acolo unde alocarea spațiului se face pe \*\*o singură axă\*\*: bare de navigație, rânduri de formular, grupuri de butoane, carduri într‑o listă care se înfășoară (wrap). Modelul mental corect este: „\*\*definim axa\*\* (orizontală/verticală), \*\*stabilim distribuția\*\* elementelor pe axa principală (`justify-content`) și \*\*alinierea\*\* pe axa secundară (`align-items`), iar `gap` controlează spațiile dintre itemi fără margini ad‑hoc”. Când apare necesitatea de alocare 2D (rânduri \*și\* coloane cu legături semantice), \*\*Grid\*\* devine alegerea naturală.

/\*\*\*\*\* Bară de navigație flexibilă \*\*\*\*\*/  
.nav{  
 display: flex;  
 flex-wrap: wrap; /\* e esențial pentru mobil: permite căderea pe rândul 2 \*/  
 align-items: center;  
 justify-content: space-between;  
 gap: .75rem;  
 padding: .75rem 1rem;  
}  
.nav\_\_brand{ font-weight: 700; }  
.nav\_\_actions{ display: flex; gap: .5rem; }

Observații cheie:  
• `flex-wrap: wrap` permite „căderea” naturală pe mobil, evitând overflow‑ul sau micșorarea agresivă.  
• `gap` în Flex este suportat modern și înlocuiește `margin` pe copil + „compensații” la capete.  
• `order` poate rearanja vizual itemii fără a schimba ordinea în DOM (folosiți cu parcimonie pentru a nu confuza ordinea focusului).

/\*\*\*\*\* Layout pe rânduri de formular \*\*\*\*\*/  
.form-row{ display:flex; flex-wrap: wrap; gap: 1rem; }  
.form-row > .field{ flex: 1 1 16rem; } /\* crește/descrescă; baza = 16rem \*/

În mod tipic, Flex rămâne excelent pentru: (1) header/nav toolbars, (2) rânduri/local alignment, (3) distribuție pe main axis cu egalizare „flex: 1” când e cazul, (4) micro‑layout interior al cardului. Dacă încercați să forțați \*\*matrici\*\* în Flex cu multe hack‑uri, este un semnal că problema e 2D și aparține \*\*Grid\*\*‑ului.

**3. CSS Grid în profunzime: matrici 2D, auto‑fit și minmax**

\*\*Grid\*\* oferă control declarativ pe \*\*rânduri și coloane\*\*. Pentru liste de carduri, grile foto, pagini cu sidebar sau machete „newspaper”, Grid reduce simțitor complexitatea. Trei idei practice:  
(1) `repeat()` pentru simetrie și economie în definirea coloanelor,  
(2) `auto-fit` / `auto-fill` + `minmax()` pentru comportament \*\*fluid\*\* fără enumerare de breakpoints,  
(3) `grid-template-areas` când vreți o hartă semantică a layout‑ului (documentează intenția direct în CSS).

/\*\*\*\*\* Card grid fluid (auto-fit) \*\*\*\*\*/  
.cards{  
 display: grid;  
 grid-template-columns: repeat(auto-fit, minmax(16rem, 1fr));  
 gap: 1rem;  
}  
.card\_\_media{ aspect-ratio: 4 / 3; object-fit: cover; }

`auto-fit` încearcă să „împacheteze” cât mai multe coloane de lățime minimă `16rem` într‑un rând, extinzându‑le uniform până la `1fr` atunci când mai există spațiu. Astfel obținem, pe mobil, 1 coloană, pe tabletă 2–3, pe laptop 3–4, fără să scriem explicit `@media` pentru fiecare treaptă. Totuși, veți apela la media queries pentru \*\*detalii\*\* (tipografie, densitate, layout al secțiunilor adiacente).

/\*\*\*\*\* Grid areas pentru pagină cu sidebar \*\*\*\*\*/  
.layout{  
 display: grid;  
 grid-template-areas:  
 "hd hd"  
 "sb main"  
 "ft ft";  
 grid-template-columns: 16rem 1fr;  
 gap: 1rem;  
}  
.header{ grid-area: hd; } .sidebar{ grid-area: sb; }  
.main{ grid-area: main; } .footer{ grid-area: ft; }

„Harta” cu `grid-template-areas` documentează intenția; colegii pot înțelege designul citind CSS‑ul, fără să deschidă Figma. Este și un instrument didactic foarte potrivit pentru proiecte educaționale: layout‑ul devine „citibil” în text.

**4. Media queries (MQ): praguri justificate de conținut**

Media queries reprezintă \*\*contractul\*\* dintre conținut și spațiul de afișare. În loc să alegem breakpoints din tabele cu „rezoluții populare”, preferăm să le \*\*deducem din conținut\*\*: „când cardurile devin prea înguste”, „când titlul sare în trei rânduri”, „când sidebar‑ul ocupă prea mult”. Practic, vă uitați la layout într‑unelte de redimensionare (DevTools), \*\*identificați momentul în care apare problema\*\*, și notați pragul. Astfel breakpoints sunt puține și motivate.

/\*\*\*\*\* Mobil -> Tabletă -> Laptop (exemplu) \*\*\*\*\*/  
@media (min-width: 480px){  
 /\* micro‑tweaks pentru small phones \*/  
}  
@media (min-width: 768px){  
 /\* 2-3 coloane, tipografie puțin mai mare \*/  
}  
@media (min-width: 1200px){  
 /\* spații mai largi, grid de 4 coloane \*/  
}

În plus, \*\*feature queries\*\* prin `@supports` pot activa variante doar când o capabilitate există (ex. un fallback când `gap` în Flex lipsește în browsere foarte vechi).

@supports (gap: 1rem){  
 .toolbar{ display:flex; gap: 1rem; }  
}  
@supports not (gap: 1rem){  
 .toolbar > \*{ margin-right: 1rem; }  
 .toolbar > \*:last-child{ margin-right: 0; }  
}

**5. Imagini responsiv și controlul raportului de aspect**

Imaginile distorsionate distrug credibilitatea vizuală. Două proprietăți esențiale:  
• `aspect-ratio` — rezervă spațiu cu un raport fix (de pildă 4:3), evitând „layout shifts” când imaginea sosește.  
• `object-fit: cover` — decupează imaginea pentru a umple containerul, fără să distorsioneze; `object-position` ajustează focalizarea vizuală.

.card\_\_media{ aspect-ratio: 4 / 3; overflow:hidden; border-radius:.5rem; }  
.card\_\_media > img{ width:100%; height:100%; object-fit: cover; object-position: 50% 40%; }

În plus, când conținutul e foarte variabil, puteți aplica \*\*trunchiere controlată\*\* (ellipsis) pe titluri și descrieri, și „expand” pe click. CSS‑ul trebuie să fie \*\*rezilient\*\* la surprizele conținutului real: nume lungi, descrieri cu termeni tehnici, apelative locale etc.

**6. Tipografie fluidă și ritm vertical**

Tipografia nu este decor, ci instrument de orientare cognitivă. Folosim `clamp()` pentru a calibra mărimi în funcție de viewport (fără a scăpa textul de sub control), iar `line-height` și spațierea verticală (margin/padding) creează \*\*ritm\*\* între blocuri. Pentru paragrafe, o măsură de \*\*45–75 caractere\*\* este, în mod tradițional, confortabilă pentru lectura continuă; pe mobile, nu forțați margini laterale exagerate — un text prea îngust obosește la fel de repede ca unul prea lung.

/\*\*\*\*\* Scară tipografică (exemplu simplu) \*\*\*\*\*/  
:root{  
 --size-body: clamp(1rem, 0.96rem + 0.3vw, 1.125rem);  
 --size-h2: clamp(1.25rem, 1.1rem + 0.8vw, 1.75rem);  
 --size-h1: clamp(1.6rem, 1.25rem + 1.8vw, 2.6rem);  
}  
body{ font-size: var(--size-body); line-height: 1.6; }  
h2{ font-size: var(--size-h2); margin: 0 0 .75rem; }  
h1{ font-size: var(--size-h1); margin: 0 0 1rem; max-width: 35ch; }

\*\*Ritm vertical\*\*: decideți o scară de spațieri (`--space-2 = .5rem`, `--space-4 = 1rem`, `--space-6 = 1.5rem` etc.) și folosiți‑o consecvent. Un ritm constant scade \*\*complexitatea vizuală\*\*, ceea ce corelează cu o \*\*experiență mai fluentă\*\* și o memorie mai bună a conținutului, concluzii susținute de studii HCI privind complexitatea și performanța utilizatorilor.

**7. Pattern‑uri mobile‑first: de la flux vertical la compoziții mixte**

Construirea layout‑ului în etape:  
(1) \*\*Flux vertical\*\* curat: conținutul se citește bine pe mobil fără MQ.  
(2) \*\*Flex în micro‑layout\*\*: bare, rânduri, grupuri.  
(3) \*\*Grid pentru matrici\*\*: carduri, galerii, pagini cu zone.  
(4) \*\*MQ pentru „corecții”\*\*: când apar rupturi vizuale, adăugați praguri justificate.   
(5) \*\*`@supports`\*\* pentru fallback‑uri curate.

/\*\*\*\*\* Flux vertical (fără MQ) \*\*\*\*\*/  
.container{ max-width: 72rem; margin: 0 auto; padding: 1rem; }  
.section + .section{ margin-top: 1.25rem; }  
.cards{ display: grid; grid-template-columns: 1fr; gap: 1rem; }  
@media (min-width: 768px){ .cards{ grid-template-columns: repeat(3, 1fr); } }

În proiecte educaționale, un \*\*pattern\*\* câștigător este „Flex în interior, Grid la exterior” (sau invers, în funcție de natură): Grid asigură structura majoră, Flex aliniază elementele interne ale cardurilor/toolbar‑urilor.

**8. Accesibilitate și ergonomie: ținte, contrast, ordine a focusului**

UI‑urile responsive nu înseamnă doar reasamblarea vizuală a componentelor; înseamnă menținerea \*\*ergonomiei\*\*.   
• \*\*Ținte suficient de mari\*\* (buton/checkbox) — ajută viteza de interacțiune (în spiritul \*\*Legii lui Fitts\*\*).   
• \*\*Decizie rapidă\*\* — reducerea numărului de alternative afișate simultan (legată de \*\*Legea lui Hick\*\*).   
• \*\*Contrast\*\* adecvat pentru text și elemente interactive — beneficii dovedite pentru lizibilitate și reținere.   
• \*\*Ordinea focusului\*\* — când rearanjați cu Flex/Grid sau `order`, asigurați-vă că ordinea de tabulare rămâne logică.

În practică: păstrați `outline` la `:focus-visible`, evitați să „ascundeți” focusul cu box‑shadow fără un echivalent la tastatură; mențineți contrast minim 4.5:1 pentru text normal; și nu supraîncărcați ecranul când treceți la viewporturi mari (panoul 4K nu înseamnă „bule de text” peste tot, ci \*\*scalare a densității\*\* și \*\*ritmului\*\*).

**9. Performanță & întreținere: simplitatea selectorilor, contain și animații sigure**

Performanța percepută este influențată de modul în care browserul recalculează stiluri, layout și pictare (paint). Deși CSS este optimizat, pot apărea penalități când:  
• Folosiți lanțuri de selectori \*\*foarte adânci\*\* (îngreunează potrivirea și întreținerea).  
• Animați proprietăți care declanșează \*\*layout\*\* (de pildă `width`, `height`); preferați `transform`/`opacity`.  
• Nu izolați sub‑arborii voluminoși; `contain: content` pe secțiuni reduce efectele colaterale ale recalculelor.

/\*\*\*\*\* Izolare de secțiuni pentru performanță \*\*\*\*\*/  
.section{ contain: content; } /\* limitează propagarea recalculelor în afara secțiunii \*/

Din perspectiva întreținerii: nume \*\*BEM\*\* pentru componente (`.card`, `.card\_\_title`, `.card--featured`), \*\*design tokens\*\* în `:root` pentru culori/spacing/typography, organizare pe fișiere (`base/`, `layout/`, `components/`, `utilities/`). Scrieți \*\*comentarii scurte, operaționale\*\*, nu eseuri în CSS — documentați intențiile și deciziile cheie.

**10. Arhitectură CSS pentru layout: BEM, utilities, tokens**

O arhitectură minimă utilă în proiecte educaționale și reale:  
• `base/` — reset/normalize pragmatice, tipografie de bază, tokens în `:root`.  
• `layout/` — Grid‑uri de pagină, secțiuni majore, media queries globale.  
• `components/` — carduri, nav, toolbars, forms; reguli locale.  
• `utilities/` — clase scurte, predictibile (`.u-margin-top-4`, `.u-hide-visually`), folosite cu moderație.  
• `themes/` (opțional) — light/dark, palete alternative pentru demo-uri.

/\*\*\*\*\* Tokens + BEM \*\*\*\*\*/  
:root{ --space-2:.5rem; --space-4:1rem; --radius:.5rem; --primary:#1a73e8; }  
.card{ border:1px solid #e5e7eb; border-radius: var(--radius); padding: var(--space-4); }  
.card--featured{ border-color: var(--primary); }  
.u-visually-hidden{ position:absolute!important; width:1px; height:1px; overflow:hidden; clip:rect(1px,1px,1px,1px); }

Principiul: \*\*puțină specificitate, multă claritate\*\*. Un set mic de utilitare bine alese și componente BEM auto‑explicative reduce dramatic complexitatea logicii CSS, ceea ce vă permite să „păstrați viteza” când apare o cerință nouă cu o oră înainte de deadline.

**11. Micro‑studiu de caz: de la mobil la panou 4K**

Să legăm toate ideile într‑o mini‑execuție. Începem cu flux vertical: titlu, scurtă prezentare, listă de carduri și o bară de filtrare minimă. Pe mobil, \*\*o coloană\*\* de carduri; la ~768 px trecem la \*\*două sau trei\*\* coloane; la ~1200 px la \*\*3–4\*\* coloane. Tipografia se dilată ușor cu `clamp()`; nav‑ul este Flex cu wrap; cardurile au media `4:3`. Pe panou 4K, \*\*nu\*\* mărim totul la dimensiuni uriașe, ci \*\*creștem spațiile\*\* (gap) și \*\*densitatea controlată\*\* (mai multe carduri simultan), menținând lizibilitatea.

/\*\*\*\*\* Consolidare (schematic) \*\*\*\*\*/  
:root{  
 --gap: 1rem;  
 --fs-body: clamp(1rem, 0.9rem + 0.5vw, 1.125rem);  
 --fs-h1: clamp(1.6rem, 1.25rem + 1.8vw, 2.6rem);  
}  
body{ font-size: var(--fs-body); }  
h1{ font-size: var(--fs-h1); }  
.nav{ display:flex; flex-wrap: wrap; gap: .75rem; justify-content: space-between; }  
.cards{ display:grid; grid-template-columns: repeat(auto-fit, minmax(16rem,1fr)); gap: var(--gap); }  
.card\_\_media{ aspect-ratio: 4/3; }  
@media (min-width: 1200px){  
 :root{ --gap: 1.25rem; }  
}

În testele vizuale veți observa cum „respiră” pagina fără salturi bruște. MQ‑urile sunt puține și motivate, Flex și Grid fac ce știu mai bine, iar unitățile fluide cu `clamp()` amortizează tranzițiile între trepte.

**12. Anti‑pattern‑uri frecvente și remedii**

1) \*\*Breakpoints „după device list”\*\*: în 2025, diversitatea ecranelor e prea mare; folosiți \*\*content‑based\*\* breakpoints.   
2) \*\*„Totul cu Flex” sau „Totul cu Grid”\*\*: alegeți instrumentul după problemă; nu forțați matrici pe Flex sau aliniere simplă pe Grid.   
3) \*\*Lanțuri de selectori adânci\*\*: creșteți specificitatea inutil; în schimb, BEM + `:where()` pentru neutralizare.   
4) \*\*Ignorarea focusului\*\*: `outline: none` fără echivalent accesibil produce bariere serioase.   
5) \*\*Animarea layout‑ului\*\*: evitați; folosiți `transform`/`opacity`.   
6) \*\*Tipografie rigidă\*\*: fără `clamp()`, veți avea fie text prea mic pe 4K, fie prea mare pe mobile.   
7) \*\*Imagini tăiate sau deformate\*\*: lipsesc `aspect-ratio` și `object-fit`.

**13. AI‑assist (Copilot/LLM) — bucla VSL pentru layout‑uri robuste**

Folosiți \*\*promturi scurte, verificări rapide\*\* (Very Short Loop). Scopul nu este să „delegăm” designul algoritmului, ci să folosim asistența pentru a explora rapid alternative și a rafina sintaxa. Exemple concrete de promturi:  
• „Propune un \*\*Grid\*\* pentru carduri cu `auto-fit` + `minmax(16rem,1fr)` și 3 breakpoints (480/768/1200).”  
• „Rescrie acest header în \*\*Flex\*\* cu wrap, acțiuni la dreapta și legături centrate pe mobil.”  
• „Calibrează o \*\*scară tipografică\*\* cu `clamp()` pentru h1/h2/h3 (range realistic).”  
• „Oferă un exemplu de `@supports` fallback când `gap` în Flex nu este disponibil.”  
• „Sugerează `tokens` (spacing, radius, culori) și aplică‑le pe `.card`, `.btn`, `.nav`.”  
După generare, \*\*evaluați critic\*\*: specificitatea, ordinea, redundanțe. Ajustați manual pentru claritate și întreținere.

**14. Checklist teoretic — ce ar trebui să stăpânești după Partea 1**

• Poți argumenta alegerea \*\*Flex\*\* vs \*\*Grid\*\* în funcție de problemă (1D vs 2D).   
• Știi să construiești \*\*grid‑uri fluide\*\* cu `auto-fit/minmax` și să aplici MQ doar ca „tuning”.   
• Poți structura \*\*media queries\*\* după \*\*conținut\*\*, nu după device, și să folosești `@supports` pentru fallback‑uri.   
• Controlezi \*\*raportul de aspect\*\* (`aspect-ratio`) și \*\*încadrarea\*\* (`object-fit/object-position`) ale imaginilor.   
• Calibrezi \*\*tipografia fluidă\*\* cu `clamp()` și menții \*\*ritmul vertical\*\*.   
• Respecți \*\*accesibilitatea\*\*: focus vizibil, contrast, ținte suficient de mari, ordine logică.   
• Înțelegi \*\*performanța\*\*: selectori simpli, `contain`, evitarea animării layout‑ului.   
• Poți descrie o \*\*arhitectură CSS\*\* simplă: BEM, utilities, tokens, fișiere organizate.

**Referințe (APA 7, cu DOI real)**

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

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