Weboldalak formázása CSS-ben

1. Alapfogalmak

(CSS, létrehozási módok, kijelölők, CSS-nyelvtan)

Bevezetés

CSS előtt:

- a HTML-dokumentumok csaknem **minden megjelenítéshez kapcsolódó része a HTML-kódon belül** volt
- a HTML-dokumentumok **bonyolultak lettek és áttekinthetetlenekké váltak**
- folyamatosan **új kimeneti eszközök** jelentek meg
- megváltozott munkaképességű emberek támogatása problémás volt

Bevezetés (folyt)

Igény:

- a felszínre került **hiányosságok pótlása**
- az addigi **szabályrendszer** következetlen-ségének megszüntetése
- weboldal-megjelenítés támogatása az **újabb eszközökön**

Megoldási módszer:

- a **tartalom** és a **formázás szétválasztása**

1. A CSS fogalma

<u>Cél</u>: a szerkezet (HTML) és a formázás (CSS) szétválasztása

CSS = Cascading Style Sheets

(egymásba ágyazott / rangsorolt / lépcsőzetes stíluslapok rendszere)

- a stílusok meghatározzák, hogy **hogyan jelenjenek meg vizuálisan** az egyes HTML-elemek
- egy **szabvány** – specifikációját a W3C felügyeli (*ld. w3schools.com*)

2. A CSS jellemzői I.

- a stílusok **a HTML** megjelenítési elemei és **attribútumai helyett** használhatók
- a stíluslapok segítségével **könnyen szét lehet választani** az oldal **tartalmát** annak kinézetétől, **megjelenésétől** (design)
- a stílusokat általában **külön állományban** (fájlban) tárolják (.css)
- a külső stíluslapokkal **gyorsítható a munka-végzés** (csak egy helyen kell változtatni)

2. A CSS jellemzői II.

- egy stíluslappal **több különböző weblap is gyorsan formázható**
- ugyanazon weboldal **többfajta eszközön való megjelenítését** is meg tudjuk adni (pl. screen, print, handheld)
- tudunk **igazodni** a weboldal megnyitását végző **kliens szg. paramétereihez**
- több stílus is hatással lehet egy elem megjelenésére – **öröklődés**

2. A CSS jellemzői III.

- a weboldalak **betöltésének ideje gyorsabbá** válik
  - kevesebb kódsor = kisebb fájlméret
  - a webböngészők a CSS-fájlokat gyakran a gyorsítótárban tárolják
  - hálózati forgalom jelentős csökkenése
- a weboldalakat **interaktívabbá tudjuk tenni** az egér és billentyűzet **eseményekre** történő stílusváltoztatással
- külső **programok** (szkriptek) **társításával** megjelenítés is **interaktívvá** tehető

3. Akadálymentesítés

- **WCAG = web content accessibility guidelines**
  - 1999 – 1.0 szabvány
  - 2008 – 2.0 szabvány
  - 2017 – 2.1 szabvány (munkaanyag)
- **információkat tartalmaz a fogyatékkal élők számára a webtartalom könnyebb eléréséhez**
- a szabvány tartalma:
  - alapelvek: észlelhetőség, működtethetőség, érthetőség, robusztusság
  - irányelvek (12)
  - teljesítési feltételek
  - elégséges és ajánlott módszerek

3. Akadálymentesítés (folyt)

- a megfelelőség szintjei (3)

A (legalacsonyabb)
ezt mindenféleképpen be kell tartani bármelyik honlapnak

AA (közepes)
magasabb szintű hozzáférhetőséget határoz meg *(közérdekű, állami/önkormányzati honlapoknál)*

AAA (legmagasabb)
nagyon komoly követelményeket támaszt a honlap készítőivel és üzemeltetőivel szemben is *(akiknek a honlapját sok sérült, fogyatékos ember látogatja)*

4. A CSS létrehozása

- a CSS stíluslapnyelv **a HTML-ben a jelölőkódokon belüli elemekhez megjelenítési tulajdonságokat rendel**
- a CSS-kódolás **többféle módon is megadható**
  - KÜLSŐ stíluslap készítése, majd a HTML-fájlhoz történő csatolása (***external stylesheet***)
  - másik webhelyről letölthetően (@import)
  - BELSŐ stílusok definiálása a HTML-dokumentumon belül (***internal stylesheet***)
  - a HTML-elemek nyitó tagjában a stílusjellemzők felsorolása (***inline style***)

4/a. External Style Sheet

A leggyakoribb megoldás:
külső fájlba (.css) mentjük el a stílusokat, majd a HTML-fájl HEAD elemében kapcsol-juk azt a weboldalhoz

```
<LINK  rel="stylesheet"
       type="text/css"
       href="… .css">
```

- rel = a két dokumentum közötti kapcsolat
- type = a csatolt külső fájl MIME típusa
- href = a külső fájl elérési útja

4/b. @import

Egy másik webhelyen definiált stíluslapfájl felhasználása (importált stíluslap):

a HEAD-ben a @import utasítással definálva "kilophatunk" egy másik weboldalról ott definiált stílusokat

```
<STYLE>
  @import url(http://www...)
</STYLE>
```

4/c. Internal Stylesheet

Egyoldalas vagy egy ***különálló weboldal*** esetén, vagy ha egy webhely egyetlen oldalán ***a külső stíluslap(ok)hoz képest kis mértékben szeretnénk változásokat elérni***, akkor a HEAD elemben definiáljuk az oldalon használandó stílusokat:

```
<STYLE>
   CSS formázás kódolása
</STYLE>
```

4/c. Internal Stylesheet (folyt)

Ha az oldalhoz van külső stíluslap is kapcsolva, akkor a belső stíluslapot célszerű a külső stíluslap hozzákapcsolását végző tagot követően definiálni:

```
<HEAD>
   <LINK rel="…" type="…" href="…">
   <STYLE>
      belső stílusdefiníciók
   </STYLE>
</HEAD>
```

4/c. Internal Stylesheet (folyt)

Ha tehát mindkét lehetőséget használjuk:

- a külső stíluslapon megjelölt, de a belsőn nem megjelölt stílustulajdonságok öröklődnek,
- a külső és a belső stíluslapon is megjelölt tulajdonságok közül a belső stíluslapon megjelölt tulajdonság lesz az érvényes,
- a belső stíluslapon megjelölt, de a külső stíluslapon nem definiált tulajdonság egyér-telműen a belsőben megjelölt formázás szerintiek alapján lesz érvényes

4/d. Inline Stylesheet

Csak speciális esetekben ajánlott ***egy HTML-címke STYLE jellemzőjének értékével*** definiálni az adott címkére vonatkozó megjelenítést (MEDIA jellemző).

```
<P STYLE="… CSS-kódok …">
```

a bekezdés szövege

```
</P>
```

Ez a formázás lényegében visszatérést jelent a HTML és a CSS szétválasztása előtti állapot irányába!

5. Kimeneti eszköz definiálása

Lehetséges a ***különböző kimeneti eszközökhez rendelt, egymástól eltérő stílusok*** meghatározása a **media** jellemző értékének megadásával:

- **all** = mindegyik eszközhöz
- **screen** = képernyőhöz
- **print** = nyomtatáshoz
- **handheld** = kézi eszközökhöz
- **projection** = kivetítőkhöz

Az egyes típusokon belül továbbpontosíthatók az egyes paraméterek!

példa - kimeneti eszköz def

Példák:

```
<LINK rel="stylesheet"
      type="text/css"
      href="….css"
      media="screen and
             (min-width: 1100px)
              and
             (max-width: 1300px">

@import url(….css) print
```

6. A CSS-rangsor

CSS = Cascading Style Sheets

cascading = lépcsőzetes / rangsorolt / egymásba ágyazott / egymásra épülő

A név utal a stílusok rangsorolt egymásra rétegződésére, ill. egymásba ágyazódására és a stílusok közötti esetleges ütközések feloldásának módjaira.

A weboldal elemeire ugyanis általában több stílusmeghatározás egyidejűleg vonatkozik!

6. A CSS-rangsor (folyt)

Rangsor:

minél közelebb van egy stílusmegadás
a formázandó elemhez,
annál nagyobb a hatása van
az adott elem megjelenésére,
tehát annál feljebb van a rangsorban,
azaz felülírja az elemtől távolabbi,
tehát a rangsorban alatta lévő stílusokat

6. CSS-rangsor (folyt)

1. böngésző alapértelmezett stíluslapjai
2. felhasználói stílusok
3. külső stílusok
4. beágyazott stílusok
5. szövegközi stílusok

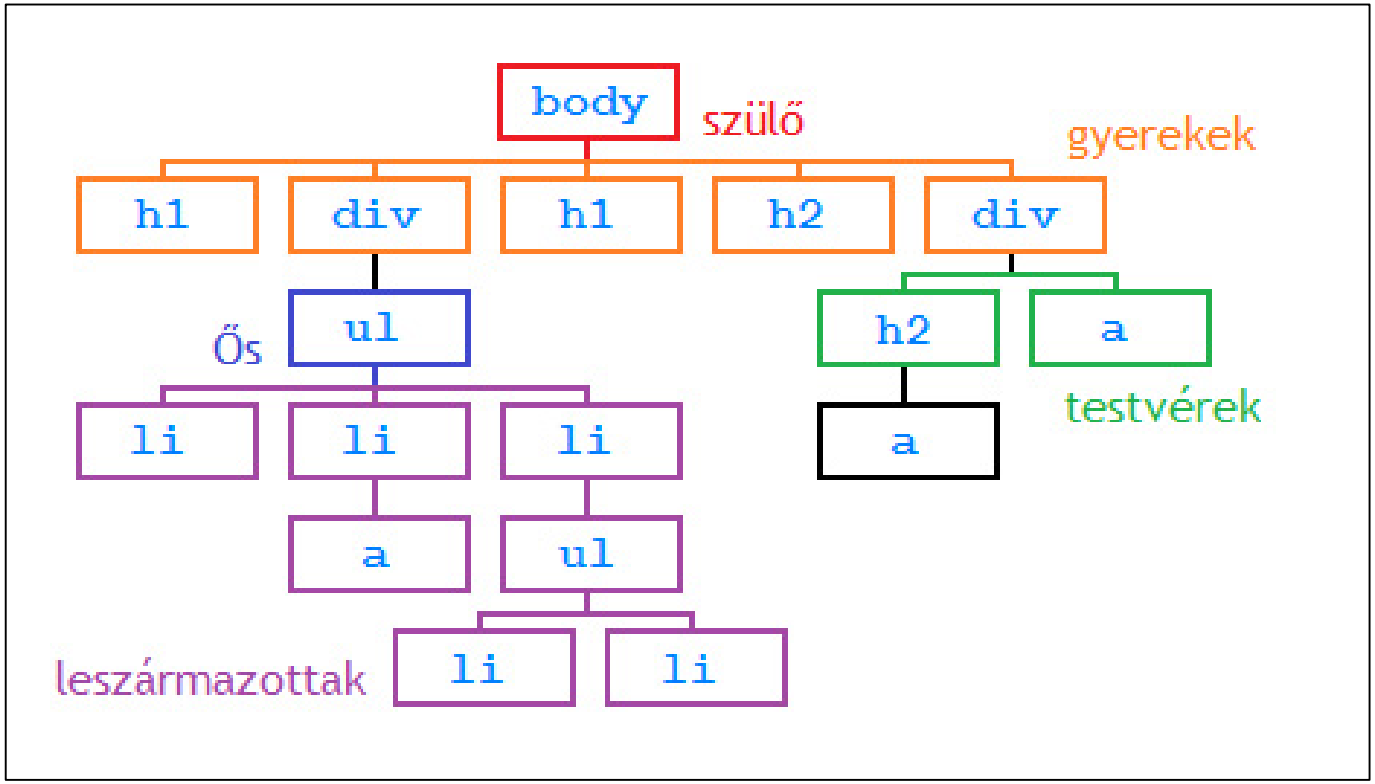
Melyik stílus fog érvényesülni, ha több stílust is definiálunk ugyanahhoz a HTML-elemhez?

Életbe lép a CSS-rangsor!

6. A CSS-rangsor (folyt)

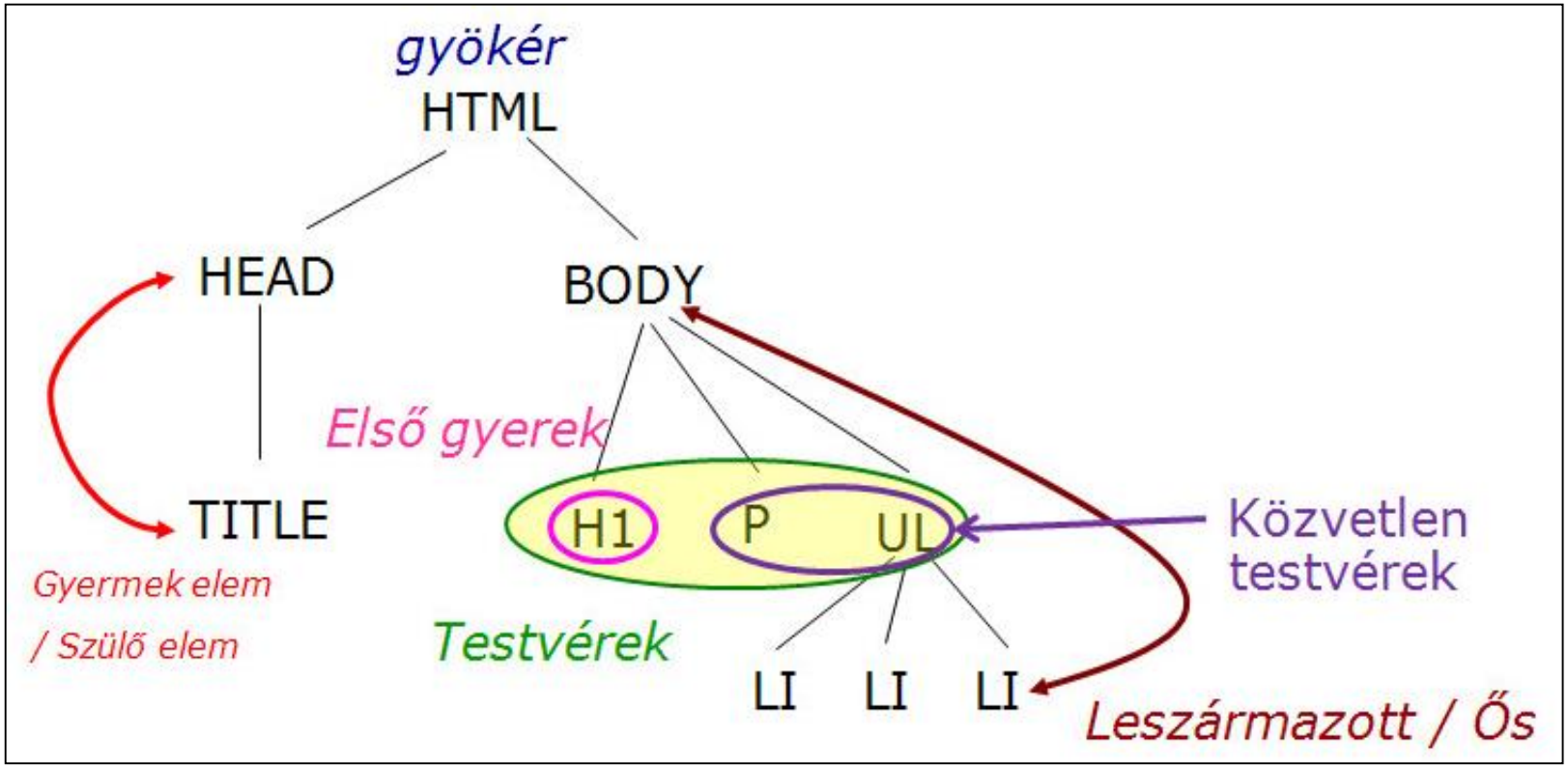
Dokumentumfa

a weboldalon elhelyezett HTML-elemek hierarchikus szerkezetű ábrázolása



6. A CSS-rangsor (folyt)

A dokumentumfában alkalmazott elnevezések



6. A CSS-rangsor (folyt)

Öröklés

ha egy elem az elrendezési struktúrában őt megelőző másik elembe ágyazódik be, akkor a megelőző elem a hierarchiában felette áll és örökíti (inherit) a stílusát a hierarchiában alatta álló eleme (feltéve, hogy arra nincs megadva külön stílus)

pl. a body örökíti tulajdonságait a teljes tartalomnak, a testvérek viszont nem örökölnek egymástól

6. A CSS-rangsor (folyt)

Szűkítés

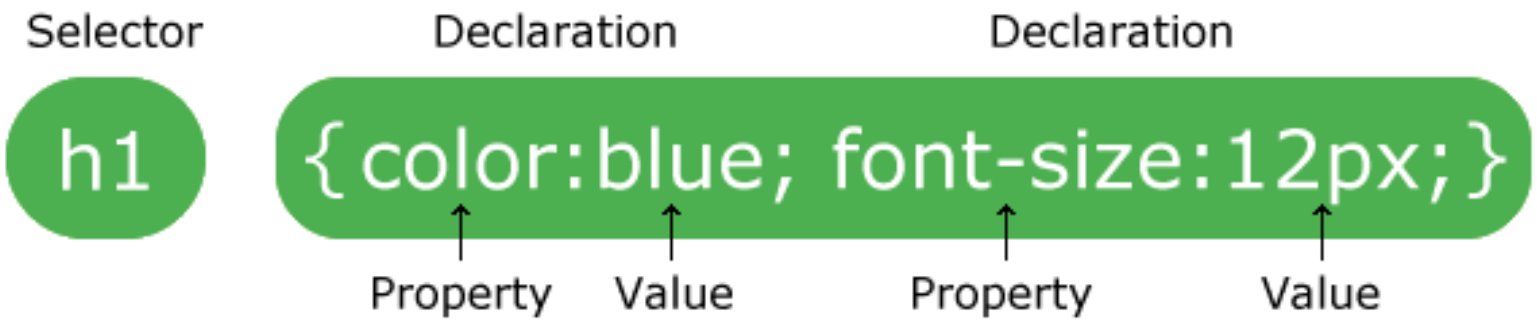
a szűkebb (pontosabb) kijelölők
felülbírálják az általánosabb kijelölőket

pl. az osztály és azonosító kijelölőknek
mindig nagyobb a súlyuk az elemkijelölőknél,
tehát felülírják
az elemkijelölőkben
meghatározott tulajdonságokat

7. CSS-nyelvtan

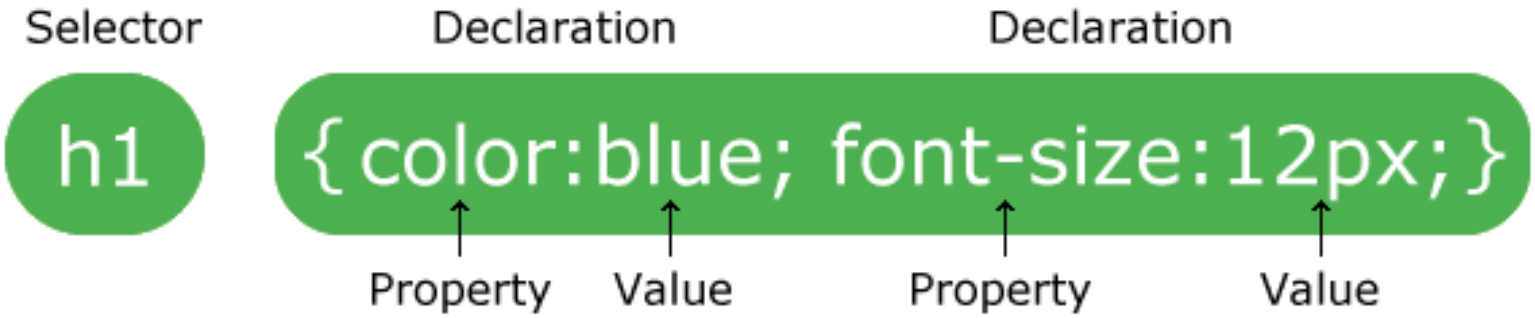
Honnan tudja a böngésző, hogy a weboldal melyik elemére és milyen formázás vonatkozik?

A stílusokat **kijelölőkkel** (kiválasztókkal, szelektorokkal) és **meghatározásokkal** lehet megadni:



7. CSS-nyelvtan (folyt)

selector = megadja, hogy a formázási utasítás a HTML melyik elemére hat



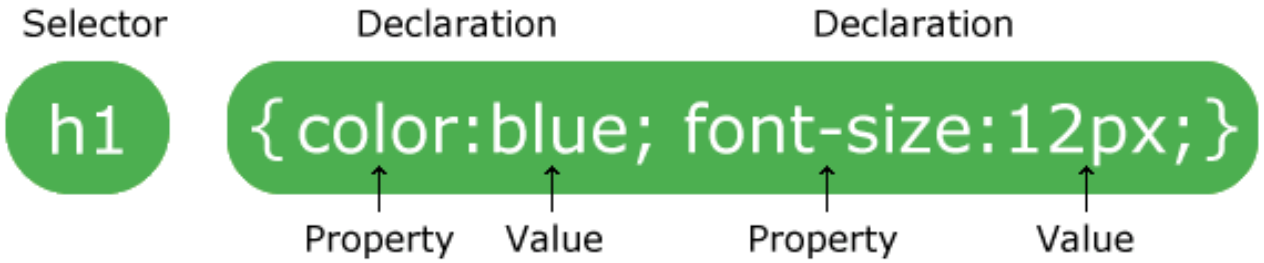
declaration = megadja, hogy hogyan nézzen ki az elem a weboldalon

- **property** = az elem azon jellemzője, amire a stílus-utasítás vonatkozik
- **value** = az adott jellemző konkrét értéke

7. CSS-nyelvtan (folyt)

Egy stílusmeghatározás formai szabálya:

kijelölő { tulajdonság-1: érték-1;
tulajdonság-2: érték-2;
…;
tulajdonság-x: érték-x; }



7. CSS-nyelvtan (folyt)

Egyes tulajdonságoknál összevontan is megadhatók az értékek: ekkor csak a gyűjtőtulajdonságot adjuk meg és hozzá soroljuk fel az értékeket.

```
div { margin-top: 10px;
      margin-right: 20px;
      margin-bottom: 10px;
      margin-left: 20px; }

div { margin: 10px 20px; }
```

7. CSS-nyelvtan - összefoglaló

Anatomy of a CSS Rule

```
h1 {
    color: orange;
    text-align: center;
}
```

Selector

Property

Value

Declaration = Property + Value

8. Mértékek és -egységek

- több tulajdonságnál is használunk **számszerű** értékeket
- a méretek megadásakor **számos mértékegység** áll a rendelkezésünkre
  (lehet: abszolút és relatív típusú)
- a **mértékegység megadása** elhagyható, ha az érték 0, egyébként **kötelező**
- a számot mindig **szóköz nélkül követi a mértékegység** (pl. 15px vagy 2em)
- ha a szám **nem egész**, akkor **tizedespontot kell használni** (pl. 1.5em)
- a tulajdonság függvénye, hogy **az érték pozitív és/vagy negatív** lehet-e

8. Mértékek és -egységek (f)

RELATÍV

- egy **másik mérethez viszonyítva** határozza meg a méretet
- lehetséges értékek:
  - **px** (pixel)
  - **em** (környezet)
  - **ex** (kis x betű)
  - **%** (másik érték arányában)

ABSZOLÚT

- minden esetben **ugyanakkora méretet jelent** a szemlélőnek
- lehetséges értékek:
  - **mm** (milliméter)
  - **cm** (centiméter)
  - **in** (inch, col, hüvelyk)
  - **pt** (pont = tipográfiai mértékegység)
  - **pc** (pica = 12 pt)

9. Színek meghatározása

- 2-féle színkeverési módszer alkalmazható

additív (összeadó):

R + G + B = fehér

RGB-kódolás

kijelzős megjelenítés

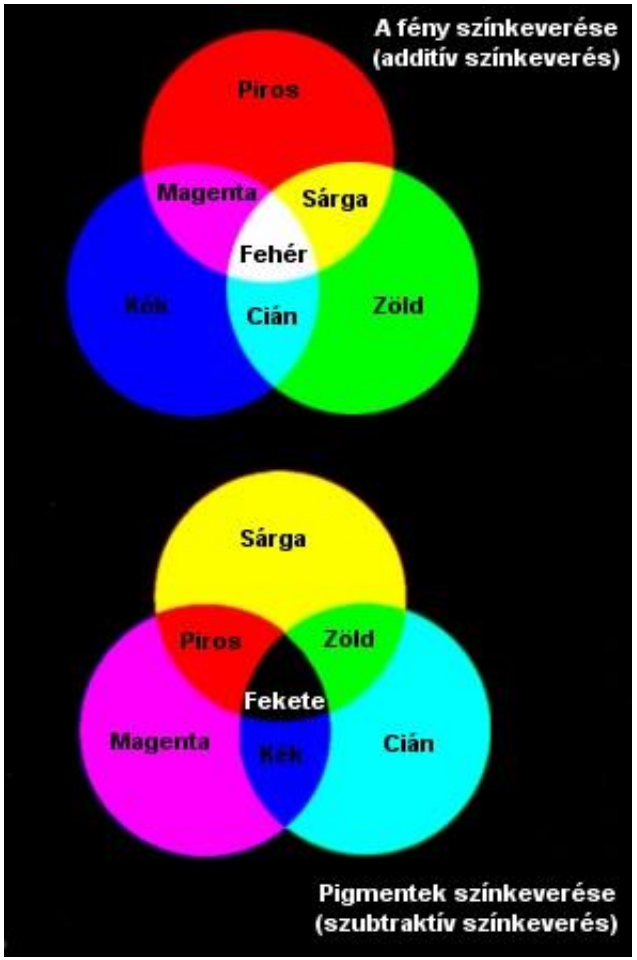
Szubsztraktív (kivonó)

C + M + Y = fekete

CMY-kódolás

(CMYK, K = black/key)

nyomdászat



9. Színek meghatározása (f)

- színmegadási módszerek

angol név (régen: 16 alapszín; ma: 147)

hexadecimális RGB-kód $\#r_1r_2g_1g_2b_1b_2$

(rövidítési lehetőség; webtűrő színek)

decimális RGB-kód rgb(rrr,ggg,bbb)

százalékos RGB-kód rgb(rrr%,ggg%,bbb%)

HSL-kód hsl(hhh,sss%,lll%)

(színárnyalat, telítettség, világosság)

RGBA-kód rgba(rrr,ggg,bbb,alpha)

(0=teljesen átlátszó, 1=átlátszatlan)

HSLA-kód hsla(hhh,sss%,lll%,alpha)

10. A kijelölők (selectors)

A **kijelölők** / kiválasztók azt definiálják, hogy **a HTML lap melyik elemére vagy elemeire vonatkoznak a meghatározásokban megadott stílusjegyek**.

- a kijelölő után egy szóközt követően, kapcsos zárójelek között szerepelnek a meghatározások ;-vel felsorolva:

```
kijelölő(k) { meghatározás(ok); }
```

Egy kijelölőhöz több meghatározás is tartozhat, ill. több kijelölőre is érvényesíthető ugyanaz a stílus.

10.1. A kijelölők fajtái

I. **egyszerű** kijelölők
(elem, azonosító, osztály, univerzális, összevonás)

II. **kombinátor** kijelölők
(az elemek közötti kapcsolat alapján: leszármazott, gyermek, szomszédos testvér, általános testvér)

III. **attribútum** kijelölők
(tulajdonság megléte és értéke alapján)

IV. **pszeudo-osztály** kijelölők
(az elemek változó állapotai szerint)

V. **pszeudo-elem** kijelölők
(az elemek tartalmának kitüntetett részei)

I. Egyszerű kijelölők

Az egyszerű szelektorokkal egyszerre nagyon sok HTML-elemet ki lehet jelölni **valamilyen közös alaptulajdonságuk** alapján:

- ugyanazon HTML-taggal hoztuk létre
- egyedi megnevezést adtunk neki
- közös nevet használunk rá
- az összes elemre érvényes legyen
- több, különböző HTML-tagra is ugyanazok a jellemzők vonatkoznak

I/a) Elemkijelölő

Az elem kijelölők *(type/element selector)* **a weblap összes azonos elemére vonatkoznak.**

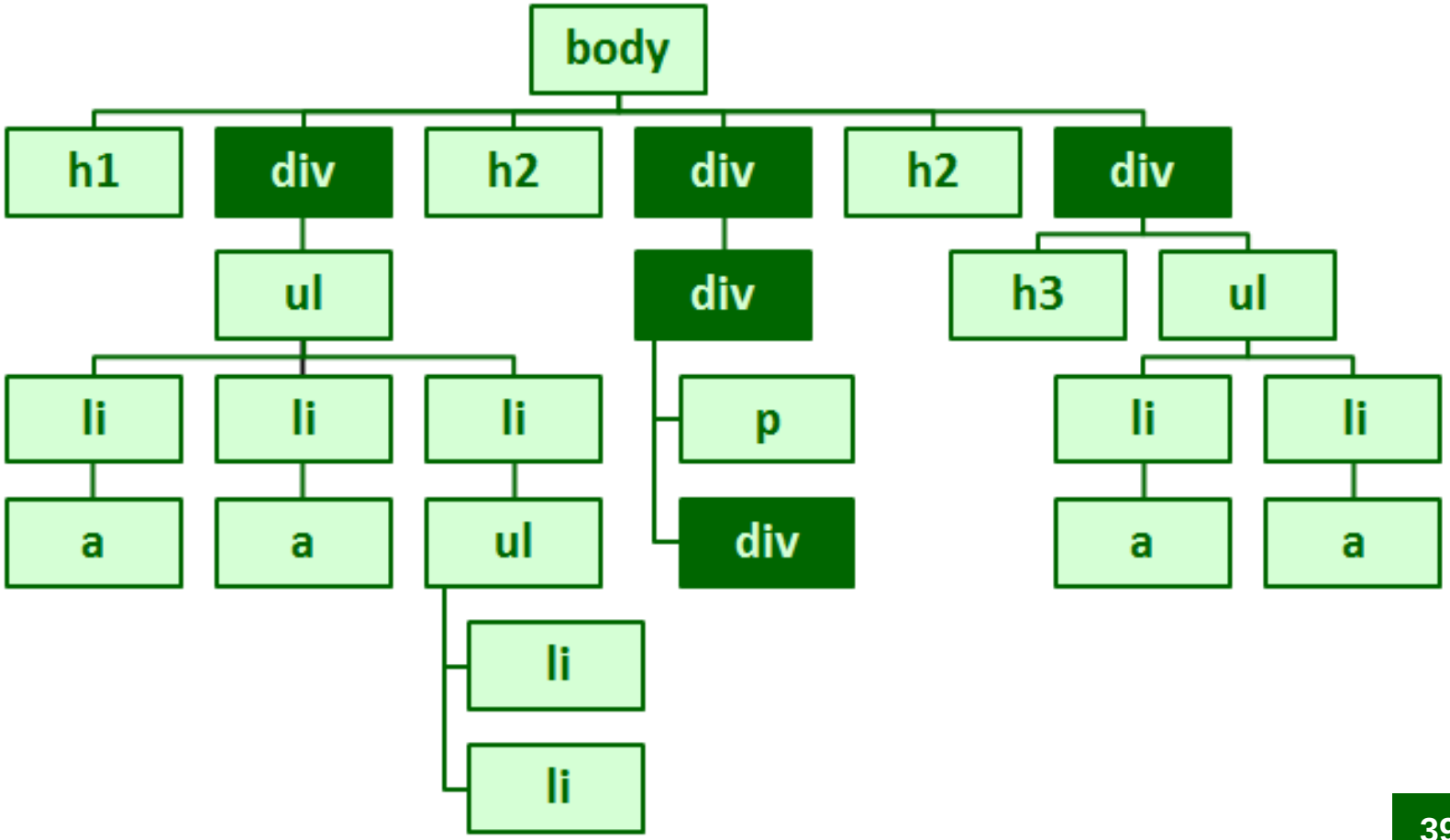
Példa*:* `p { … }`

Ekkor az összes, a dokumentumban megtalálható ugyanolyan taggal ellátott elemre érvényesülni fog a formázás.

Alkalmazásának célja: alapbeállítások megadása *(A HEAD-ben lévő elemek kivételével az összes tagra használható.)*

példa elemkijelölőre

```
div { …}
```



I/b) Azonosító kijelölő

Az azonosító kijelölők *(ID selector)*
a HTML-dokumentum egy egyedi ID jellemzővel ellátott helyére vonatkoz-nak.

Példa: *HTML:* `<p id="nev"> … </p>`

CSS: `#nev { … }`

Akkor alkalmazzuk, ha a weboldal egyetlen konkrét helyén szeretnénk formai változásokat megvalósítani.

I/c) Osztály kijelölő

Az osztály kijelölők *(class selector)*
az általunk ugyanabba az osztályba sorolt elemek azonos megjelenítését határozzák meg.

Példa: *HTML:* `<p class="o"> … </p>`

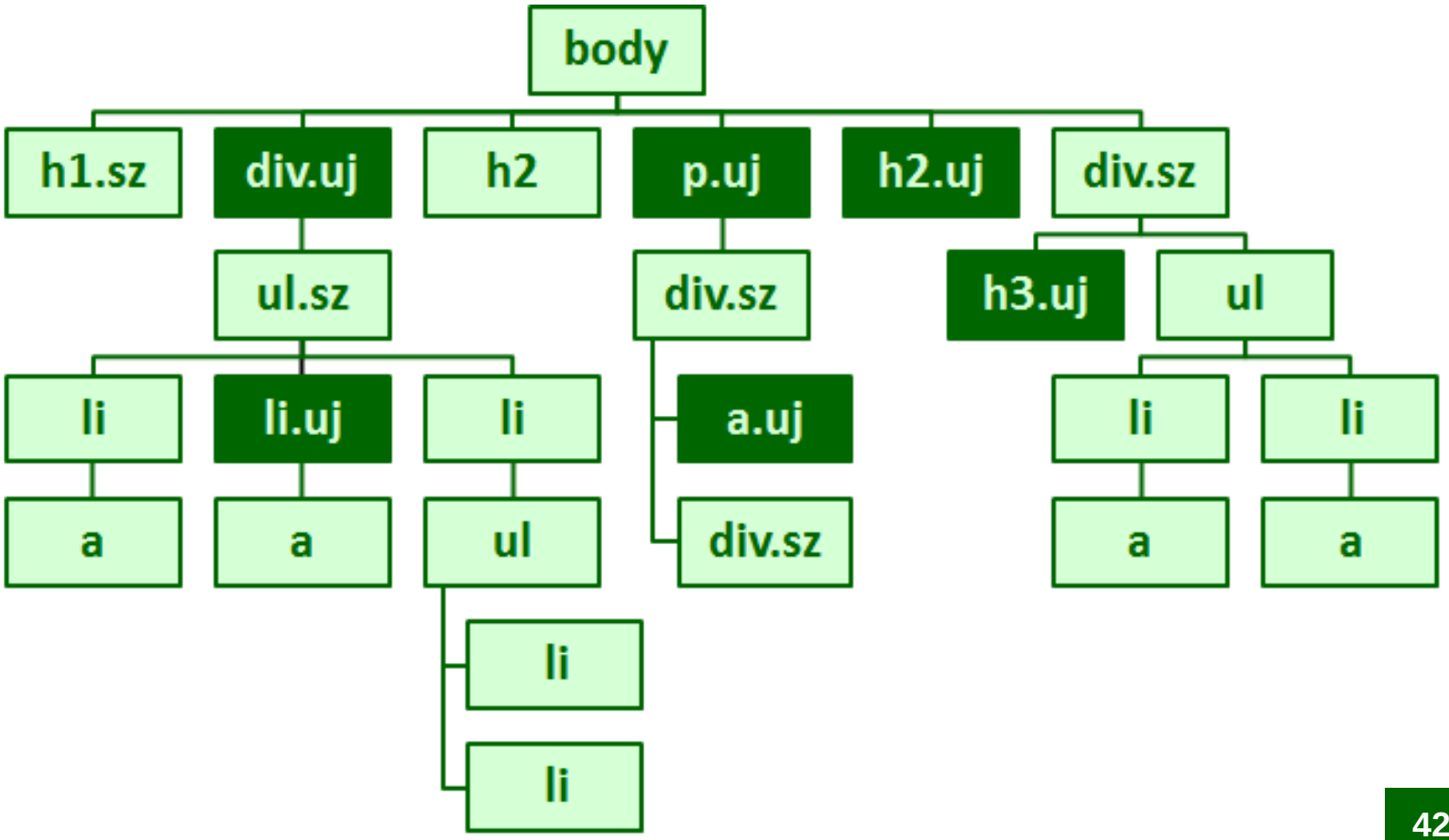
CSS: `.o { … }`

Ugyanahhoz az elemhez több osztálymegadás is tartozhat, de ekkor elegendő felsorolni a neveket:

```
<div class="nev1 nev2 nev3">…</div>
```

példa osztály kijelölőre

```
.uj { … }
```



I/d) Univerzális kijelölő

Kevéssé használt kijelölő a * *(universal selector)*, **amely a dokumentumban lévő minden egyes elemet megjelöl.**

```
*  {  …  }
```

*A :not(…) segítségével a zárójelben lévő elem kijelölését lehet letiltani, azaz a *:not(…) használatával a zárójelben megadott elem kivételével minden kijelölhető:*

```
*:not(p)  {  …  }
```

I/e) Csoport kijelölő

A csoport kijelölés *(group of selectors)* **több elemhez rendeli ugyanazt a stílust.**

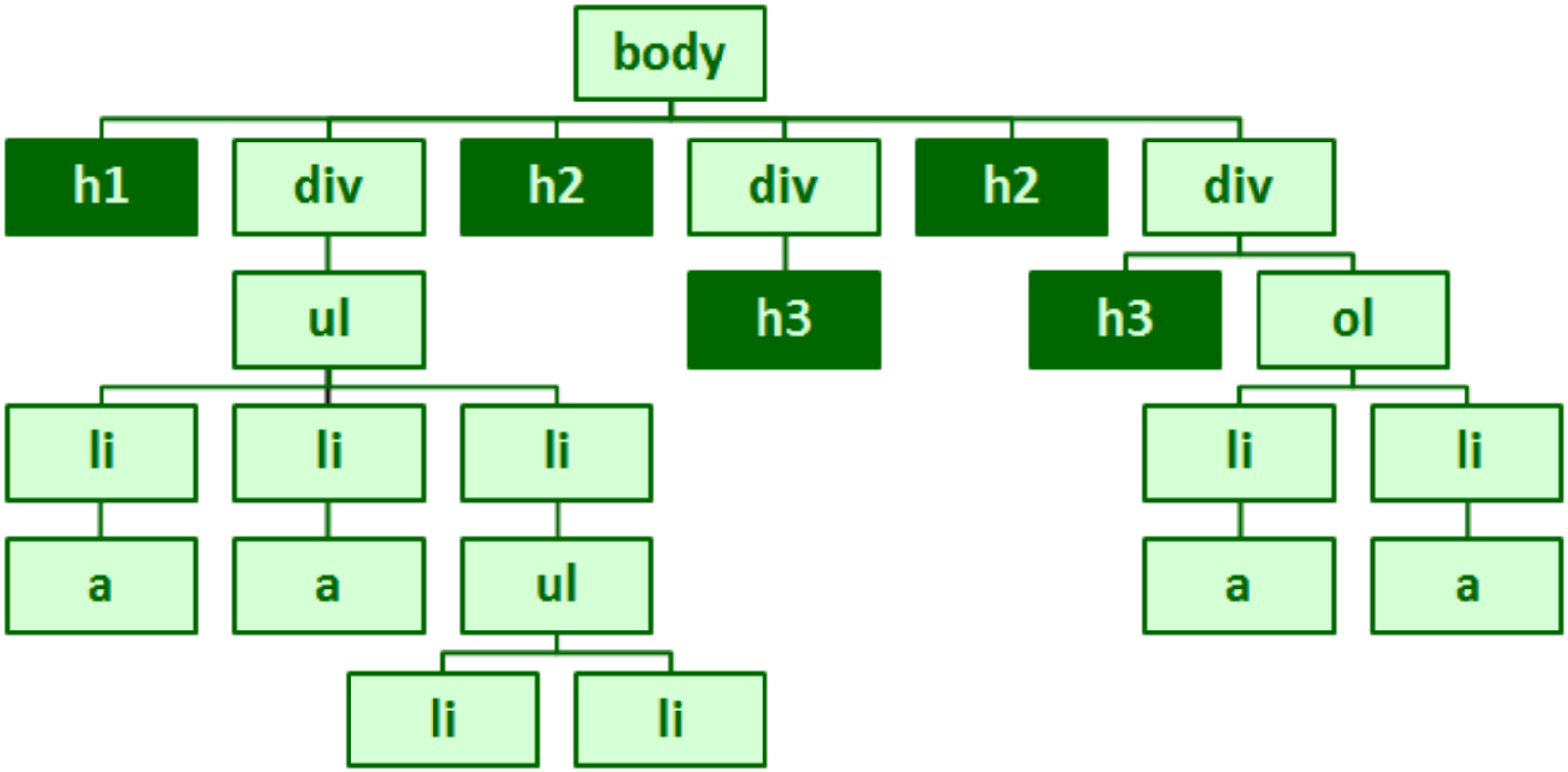
A kijelölt elemeket vesszővel kell elválasztani egymástól és a vessző után szóközt kell írni.

Példa: `p, h1, div { … }`

Segítségével az azonos jellemzőket együtt adhatjuk meg, így a kódsorok száma kevesebb lehet.

példa csoport kijelölőre

```
h1, h2, h3 { … }
```



II. Kombinátor kijelölők

A **kombinátorok** *(combinators)* esetén a kijelölés **a HTML-elemek közötti kapcsolat** alapján történik, azaz **az öröklési viszonyok**, vagyis a családfa mintáját követő dokumentumfa alapján történő kijelölést tesznek lehetővé**.**

- leszármazott
- gyermek (elsőszintű leszármazott)
- szomszédos testvér
  (közös szülő + egymást követő elemek)
- általános testvér (közös szülő)

II/a) Leszármazott kijelölő

Egy HTML-elem leszármazottjainak *(descendant)* elemnevét **szóközzel elválasztva** adhatjuk meg.

```
szelektor1  szelektor2  szelektor3
      { stílusdefiníció }
```

(A formázás mindig az utolsó elemre vonatkozik!)

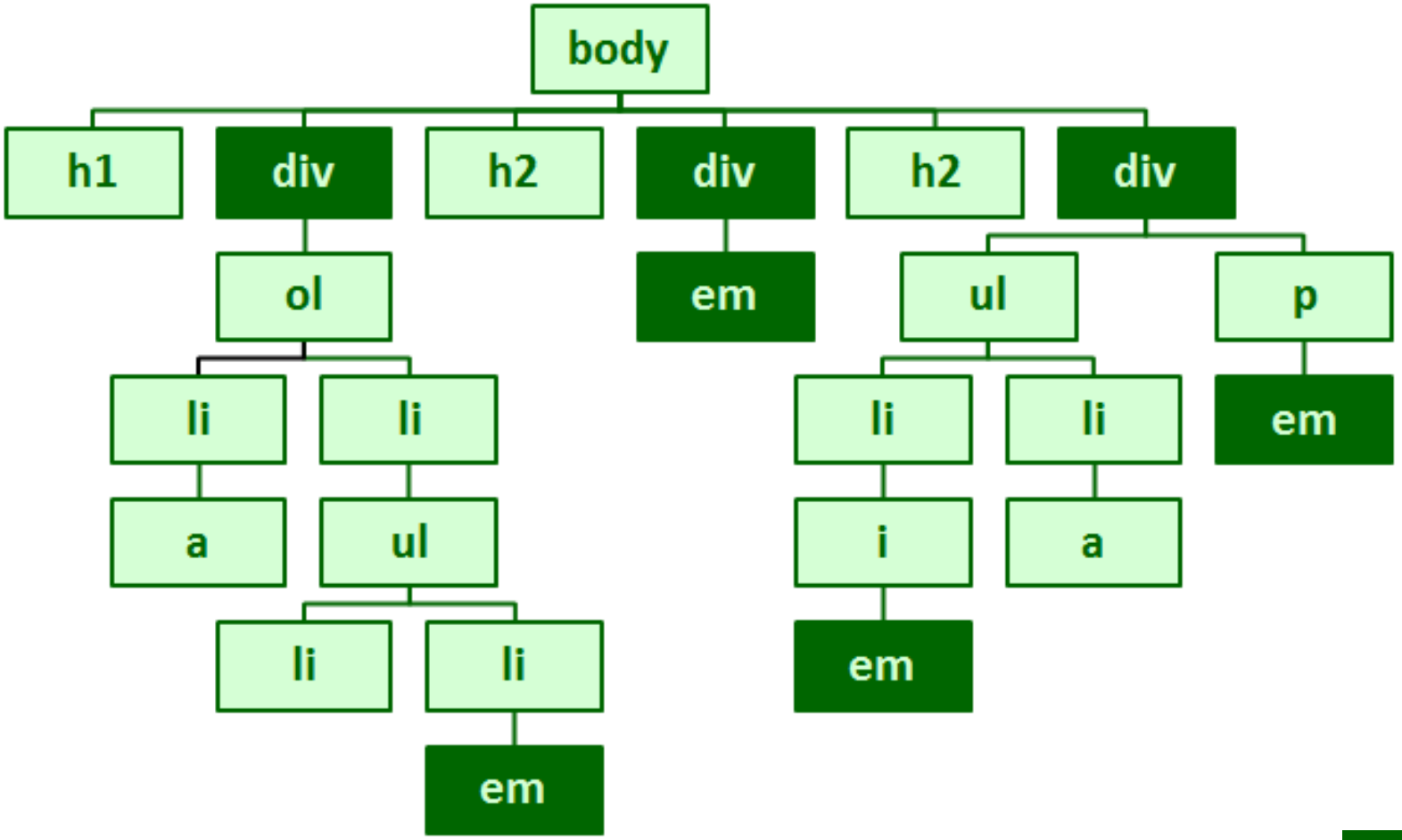
Példa*:*

```
#szak1 p { … }
.kiemelt div { … }
p .vege { … }
#fej p.bevezetes { … }
ul li li { … }
```

Akkor alkalmazzuk, ha egy kijelölést pontosítani szeretnénk.

példa leszármazott kijelölőre

```
div em { … }
```



II/b) Gyermek kijelölő

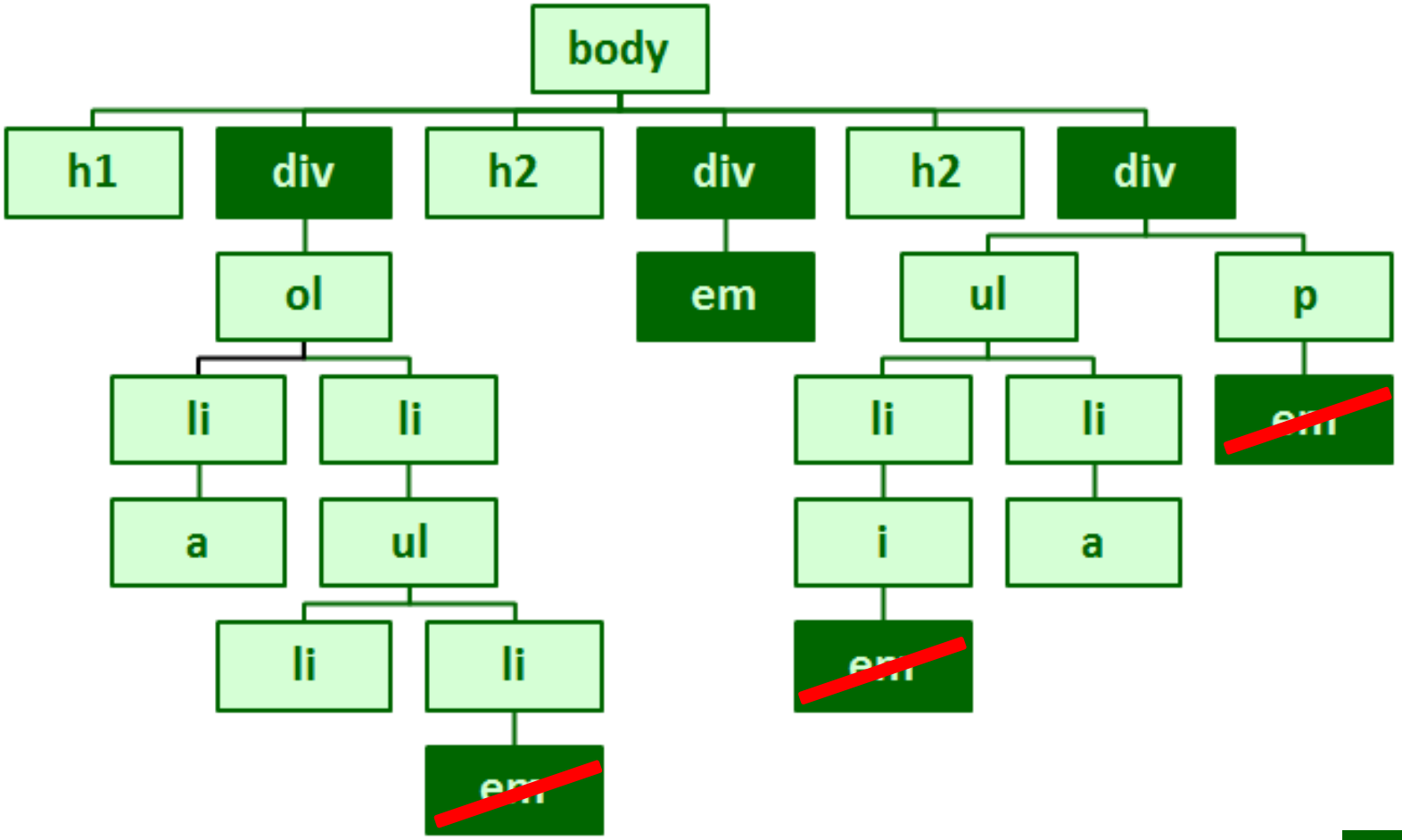
Két elem közötti **szülő-gyerek viszonyt** a gyermek kijelölő *(child selector)* írja le. **Jele a >,** amelyben a bal oldalon a szülő, a jobb oldalon a gyermek elem található.

```
szelektor1 > szelektor2
{ stílusdefiníció }
```

pl. `body > p` jelöli a body tag p bekezdéseit (tehát csak azokat a bekezdéseket, amelyek közvetlen leszármazottjai (gyermekei) a body-nak)

példa gyermek kijelölőre

```
div > em { … }
```



II/c) szomszédos testvér kij.

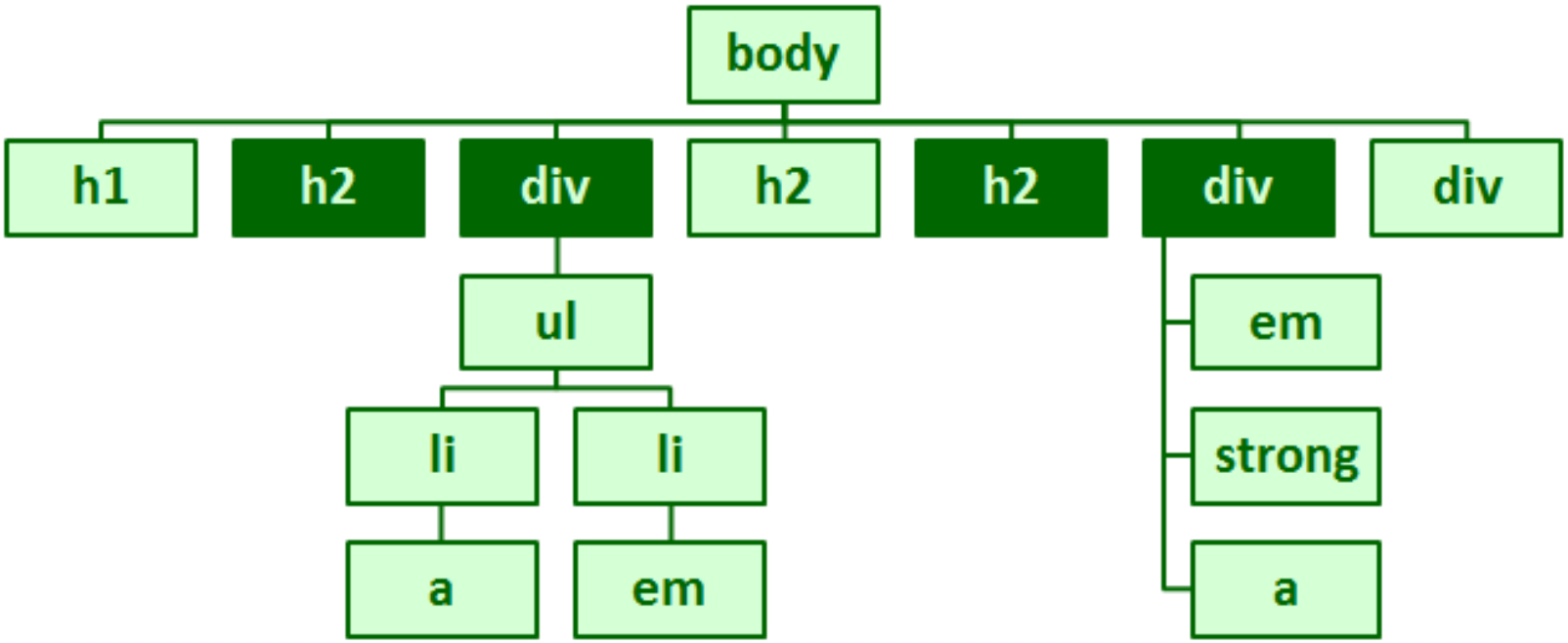
Két elem közötti testvéri viszonyt ír le
(tehát közös a szülő, de az elemek egymásnak nem leszármazottjai).

szomszédos testvér, jele a +
az első elem a sorban közvetlenül megelőzi a másodikat és közös a szülőjük

pl. `h1.piros + p.kek` jelöli a "piros" osztályú h1 címet közvetlenül követő "kek" osztályú bekezdés formázása

példa szomszédos testvérre

```
h2 + div { … }
```



II/d) testvér kombinátorok

általános testvér, jele a ~

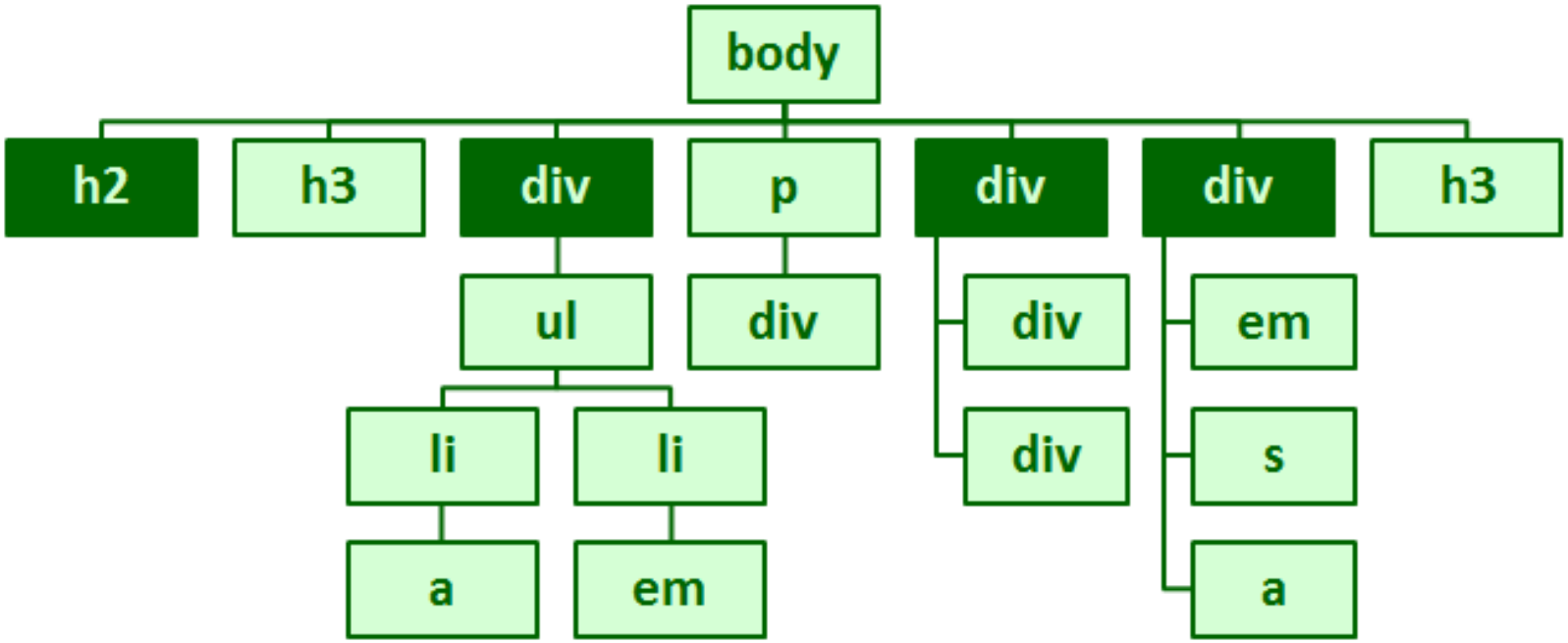
az első elem a sorban nem feltétlenül közvetlenül előzi meg a másodikat és közös a szülőjük

pl. `h1 ~ p` a h1 címet valahol később követő bekezdéstestvér jelöli

(a kettő között akár más olyan tagok is lehetnek, amelyeknek a szülője megegyezik a h1 és a p tag szülőjével)

példa általános testvérre

```
h2 ~ div { … }
```



III. Attribútum kijelölők

Az **attribútum** *(attribute)* alapú kijelölésnél a kiválasztást annak alapján szabályozzuk, hogy az egyes elemek **milyen tulajdonságokkal rendelkeznek és azok milyen értékeket vesznek fel**.

```
szelektor[tulajdonság=érték]
```

(a megadott tulajdonság értékére különböző szűréseket végezhetünk:
pontos egyezés, részegyezés, eleje-vége egyezés)

III/a) Egyszerű jellemzőkijelölő

Rendelkezik az adott HTML-elem a meghatározott tulajdonsággal, azaz megadtuk az elemnél a tulajdonságot:

```
tagnév[tulajdonságnév] { … }
```

`div[title]` azokat a blokkszintű elemeket jelöli ki, amelyeknek megadtuk a title tulajdonságát

`img[width]` azokat a képeket jelöli ki, amelyeknek van szélességérték beállítva a felhasználó részéről

III/b) Pontos jellemzőérték

Rendelkezik az adott HTML-elem a meghatározott tulajdonsággal és pontosan a megadott értékkel:

```
tagnév[tulajdonság=érték] { … }
```

`input[type=text]` csak az egysoros szöveges beviteli mezőkre vonatkozik a kijelölés

`img[width="200px"]` azokat a képeket jelöli ki, amelyeknek a szélessége pontosan a 200px érték

III/c) Részleges jellemzőérték

Rendelkezik az adott HTML-elem a meghatározott tulajdonsággal és a megadott értékkel mint szóközökkel tagolt kifejezéssel:

```
tagnév[tulajdonság~=érték] { … }
```

`a[title~="SZTE"]` olyan linkek, amelynek a magyarázó szövegében (title) benne van az SZTE mint önálló szó
(nem jó: SZTE-tag, SZTEs)

III/c) Részleges jellemzőérték

Rendelkezik az adott HTML-elem a meghatározott tulajdonsággal és a keresett értéket kötőjelek is határolhatnak:

```
tagnév[tulajdonság|=érték] { … }
```

`p[lang|="en"]` olyan bekezdések, amelynek a nyelvi beállítása az angol (en) valamelyik változata *(pl. en-us, en-gb)*

III/d) Jellemzőérték eleje

Rendelkezik az adott HTML-elem a meghatározott tulajdonsággal és az értékének kezdete a megadott.

```
tagnév[tulajdonság^=érték] { … }
```

`div[class^="fej"]` olyan blokkok, amelynek az osztályneve a „fej” kifejezéssel kezdődik
(pl. fejezet, fej1, fejelés, fejjelemzo_3,…)

III/e) Jellemzőérték vége

Rendelkezik az adott HTML-elem a meghatározott tulajdonsággal és az értékének vége a megadott.

```
tagnév[tulajdonság$=érték] { … }
```

`div[id$="teszt"]` olyan blokkok, amelynek az azonosítóneve a „teszt” kifejezésre végződik *(pl. elsoteszt, vegteszt, kor_teszt)*

III/f) Jellemzőérték részlete

Rendelkezik az adott HTML-elem a meghatározott tulajdonsággal és az értékében szerepel a megadott.

```
tagnév[tulajdonság*=érték] { … }
```

`p[id*="resz"]` olyan bekezdések, amelynek az azonosítóneve tartalmazza a „resz” kifejezést *(pl. reszlet, elsoresz, kireszletezo, …)*

IV. Pszeudo-osztály kijelölők

A **pszeudo-osztály** (ál-osztály, látszólagos) kijelölők *(pseudo-class selectors)*
az egyes elemeknek
bizonyos állapotukban való elérését
teszik lehetővé.

```
szelektor:pszeudo-osztály
{ stílusdefiníció }
```

Önmagában sosem fordul elő és mindig valamilyen elemhez kapcsolódik!

két fő típusa létezik:
dinamikus és strukturális pszeudo-osztály jelölők

IV/a) Dinamikus ál-osztályok 1.

A link típusú ál-osztály kijelölők a hivatkozások formázásánál kapnak kiemelt szerepet.

- a még ***meg nem látogatott*** hivatkozások formázásához: **`a:link { … }`**
- a ***felkeresett, meglátogatott*** hivatkozások formázásához: **`a:visited { … }`**

Példa*:*

```
#nav a:visited { … }
#container:link { … }
```

IV/a) Dinamikus ál-osztályok 2.

Bizonyos ál-osztály kijelölők az elemek egyes dinamikusan változó állapota alapján jelölik ki a formázandó részt.

- ha ***rámutatunk az egérrel*** az adott elemre:
  `div:hover { … }`
- ha ***lenyomva tartjuk az egérgombot***:
  `button:active { … }`
- ha ***az elem a fókuszba kerül***:
  `input:focus { … }`

IV/a) Dinamikus ál-osztályok 3.

A hivatkozásoknál az ugrási célpont formázásához használható.

```
:target { … }
```

pl. `:target { border: 2px solid #D4D4D4; }`

```
<p><a href="#cel1">Ugrás az 1. célhelyre</a></p>
<p><a href="#cel2">Ugrás a 2. célhelyre </a></p>
<p> szöveg……</p>
<p id="cel1"><b>1. célhely tartalma</b></p>
<p id="cel2"><b>2. célhely tartalma</b></p>
```

Amikor rákattintunk valamelyik linkre, a célhelyet a megadott módon fogja formázni.

IV/a) Dinamikus ál-osztályok 4.

Űrlapoknál az egyes elemek alkalmaz-hatóságát szabályozhatja.

```
:disabled { … }
```

pl. `input[type=number]:disabled { … }`
a letiltott numerikus egysoros beviteli űrlapmezők esetén lehetőséget biztosít az eltérő formai megjelenítésre *(pl. szürke színnel jelenjenek meg)*

```
:enabled { … }
```

pl. `input[type=text]:enabled { … }`
az engedélyezett (írható) szöveges egysoros beviteli űrlapmezők formázáshoz ad információkat

IV/a) Dinamikus ál-osztályok 4.

Űrlapoknál a bejelölt választógomb (rádiógomb) vagy jelölőnégyzet (checkbox) formázásához használható.

```
:checked { … }
```

pl. `input:checked`

```
{ height: 50px; width: 50px; }
```

az összes input tag bejelölt állapotú elemének méretét beállítja 50x50 képpont nagyságúra

A dinamikus kijelölők tehát azért ál-osztályok, mert állapotuk nem a HTML-kódtól, hanem a weboldal látogatójának tevékenységétől függ

IV/b) Lang-típusú ál-osztályok

Az adott elemre beállított nyelv szerinti választáshoz használt kijelölő.

```
:lang(nyelv) { … }
```

pl. `p:lang(it)`

```
{ background: yellow; }
```

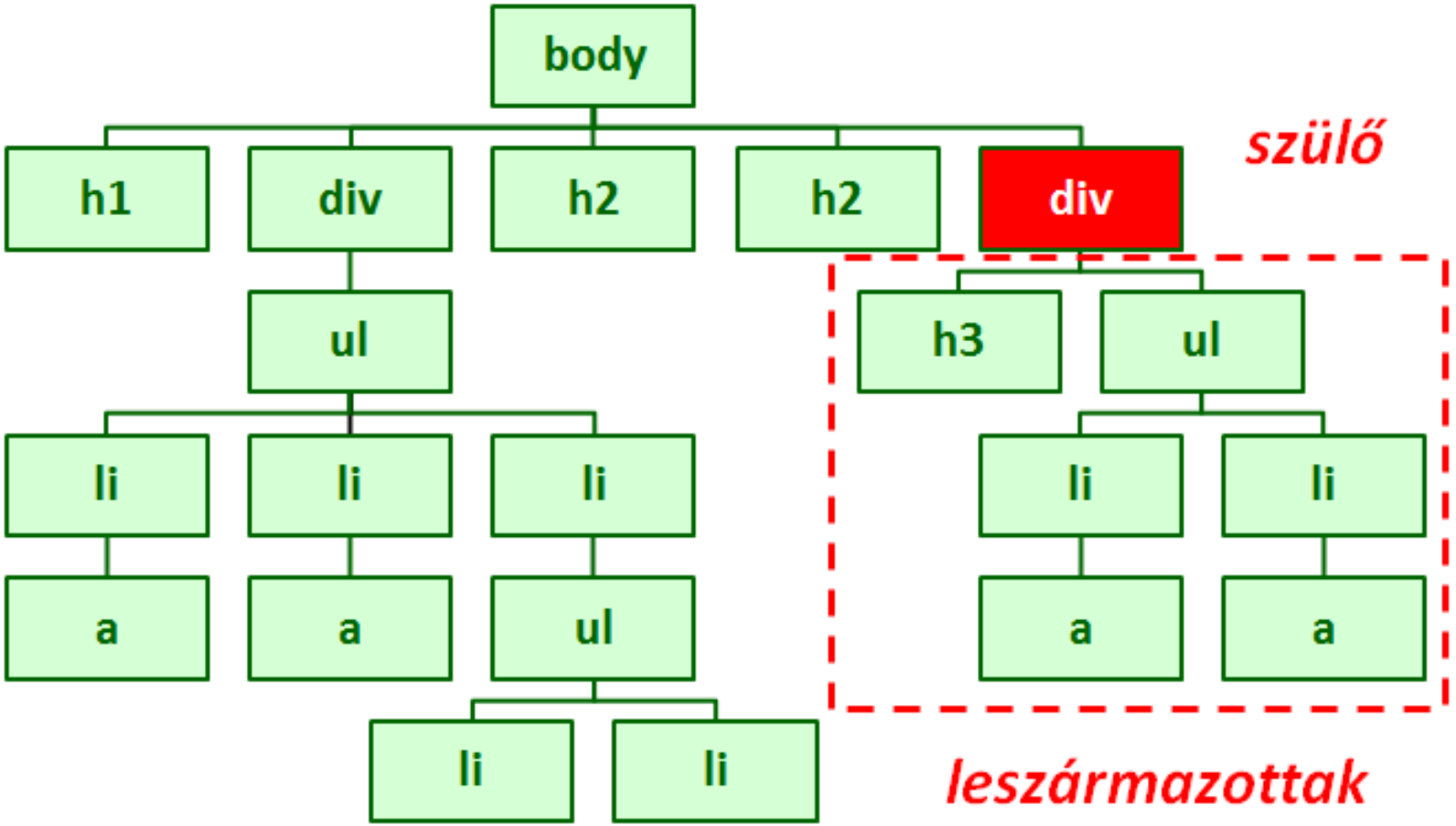
az olasz nyelvűre beállított bekezdéseket formázza meg, azaz ott, ahol a <p> HTML-tag a lang=it beállítást tartalmazza
(pl. <p lang=it>)

IV/c) Strukturális ál-osztályok

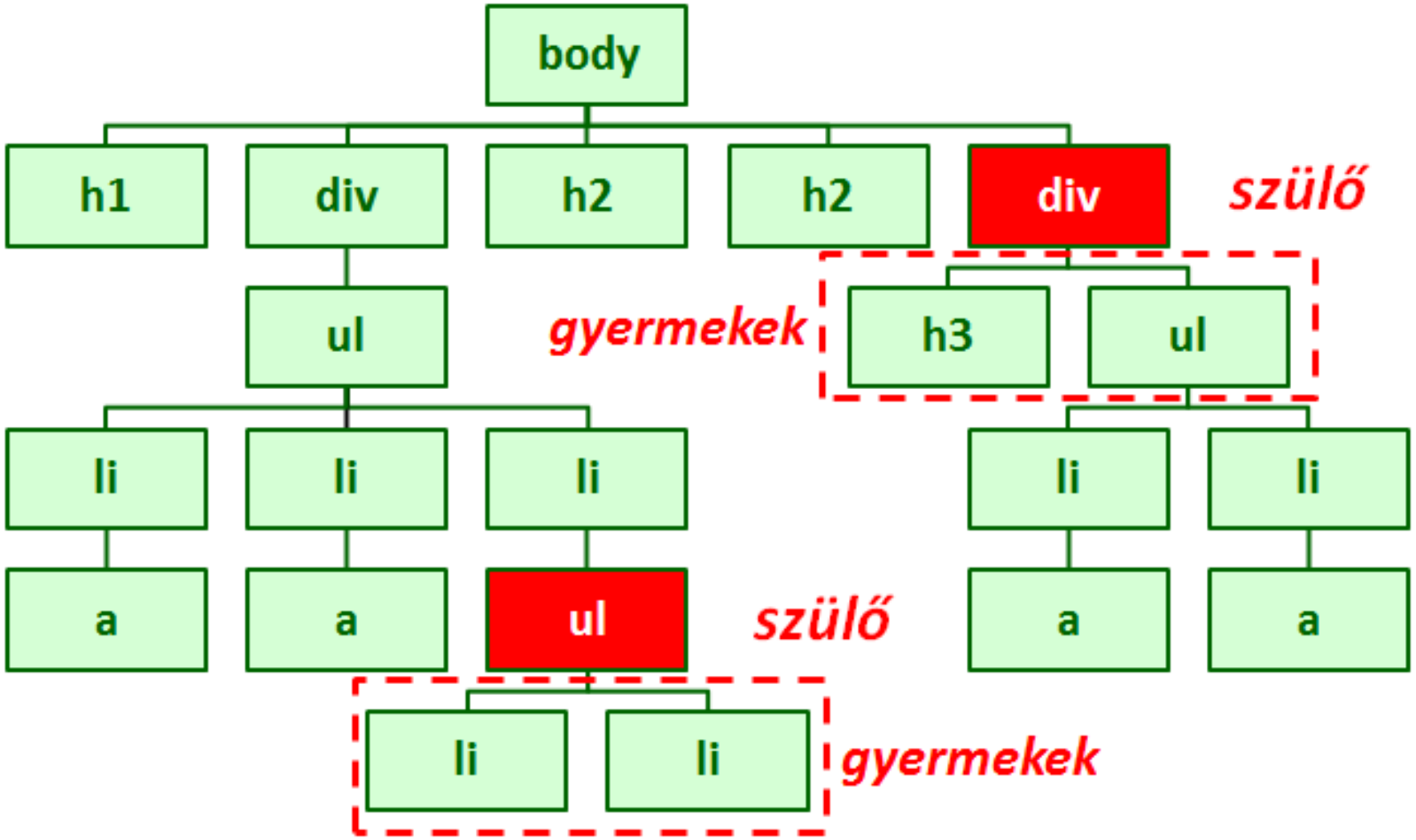
A **strukturális ál-osztályok** lehetővé teszik, hogy a dokumentumfa által tartalmazott **strukturális információk felhasználásával** a más kijelölőkkel egyszerű módon el nem érhető kiválasztást végezhessünk.

pl. gyökérelem, első elem, utolsó elem
valahanyadik gyermek, gyermekek
valahanyadik testvér, testvérek

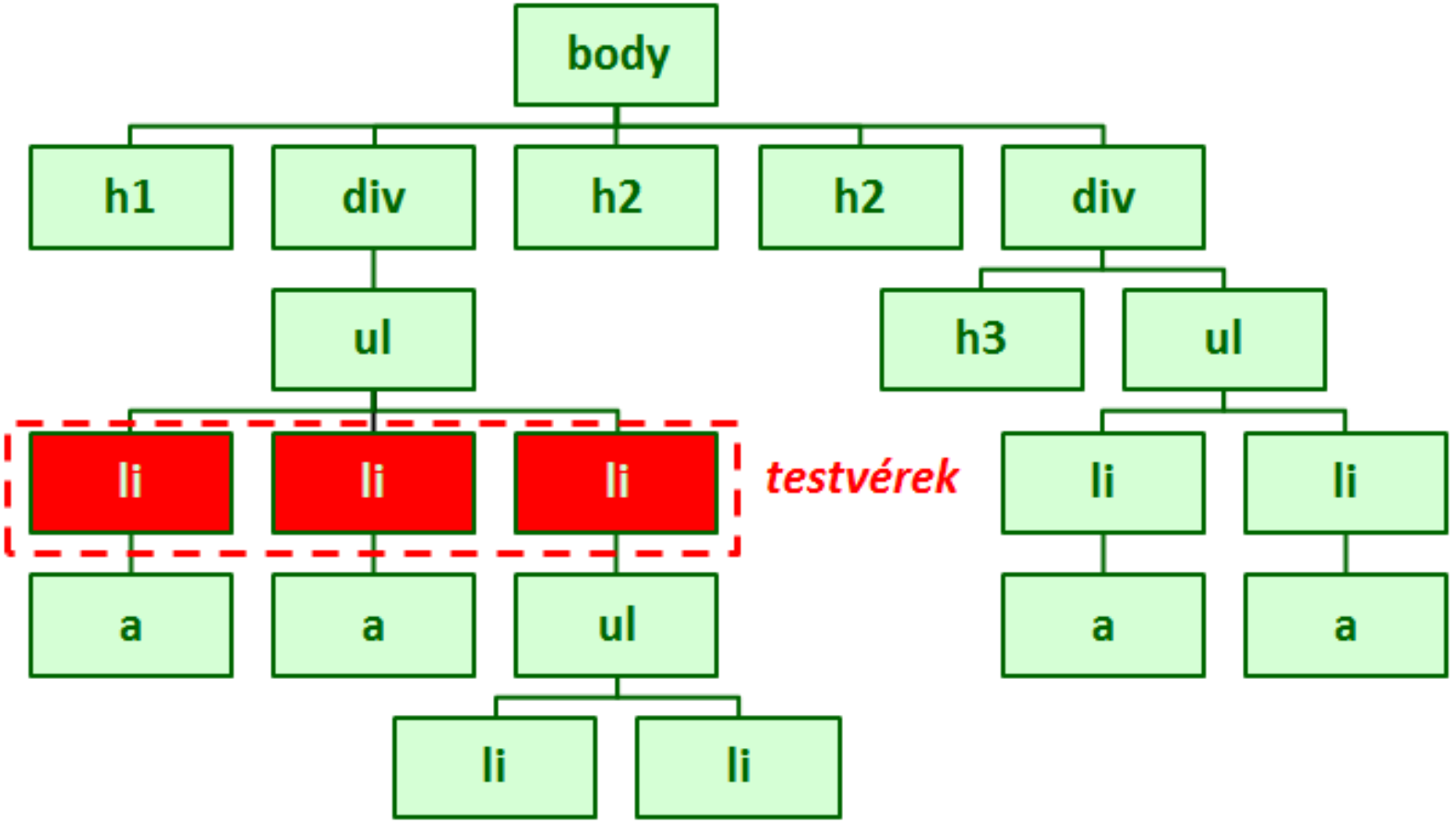
a dokumentumfa elemei



a dokumentumfa elemei



a dokumentumfa elemei



IV/c) Strukturális ál-oszt. 1.

`:root`

a dokumentum gyökér-elemét (html) képviseli

`:empty`

gyermektelen elem formázásához (levél)

Mi a különbség?

`div:empty` `div :empty`

IV/c) Strukturális ál-oszt. 2.

`tagnév:nth-child(x)`

a tag elem valahanyadik gyermeke egy másik (beágyazó) elemnek

pl. **`p:nth-child(6)`** egy tetszőleges elem bekezdésgyermekei között a 6. tagot jelöli meg

az "x" értéke lehet: 0, egy pozitív szám, "an+b" alakú szám (3n+1), odd (páratlan), even (páros)

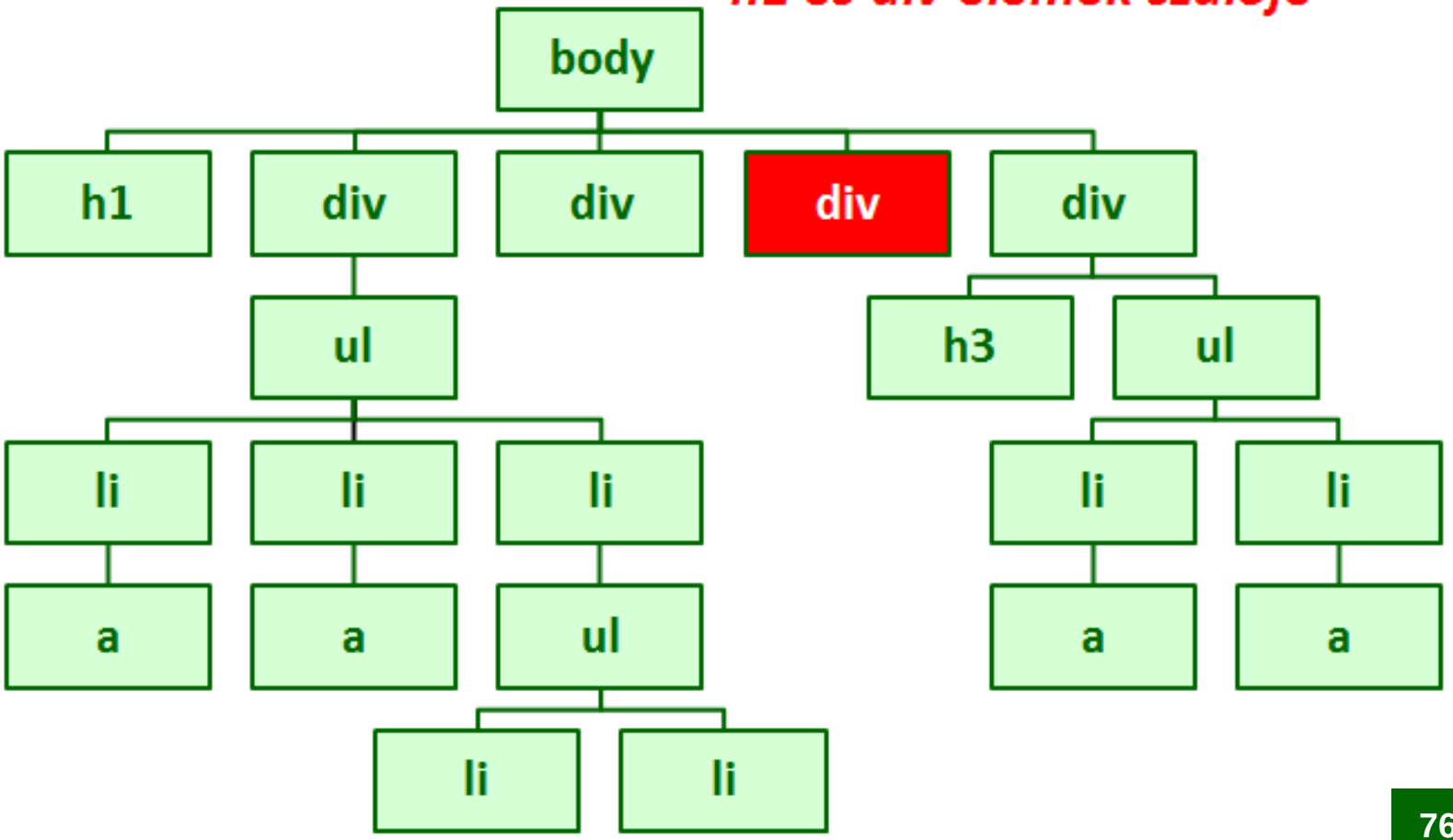
Mit jelent?

```
body :nth-child(4)
li:nth-child(3n)
```

példa dokumentumfával

div:nth-child(3)

h1 és div elemek szülője



IV/c) Strukturális ál-oszt. 3.

pl. *a páratlan bekezdések formátuma:*

```
p:nth-child(odd) { background: red; }
```

a páros bekezdések formátuma:

```
p:nth-child(even) { background: blue; }
```

Az első bekezdés.	→ piros
A második bekezdés.	→ kék
A harmadik bekezdés.	→ piros
A negyedik bekezdés.	→ kék
Az ötödik bekezdés.	→ piros
A hatodik bekezdés.	→ kék
A hetedik bekezdés.	→ piros

IV/c) Strukturális ál-oszt. 4.

`tagnév:first-child`

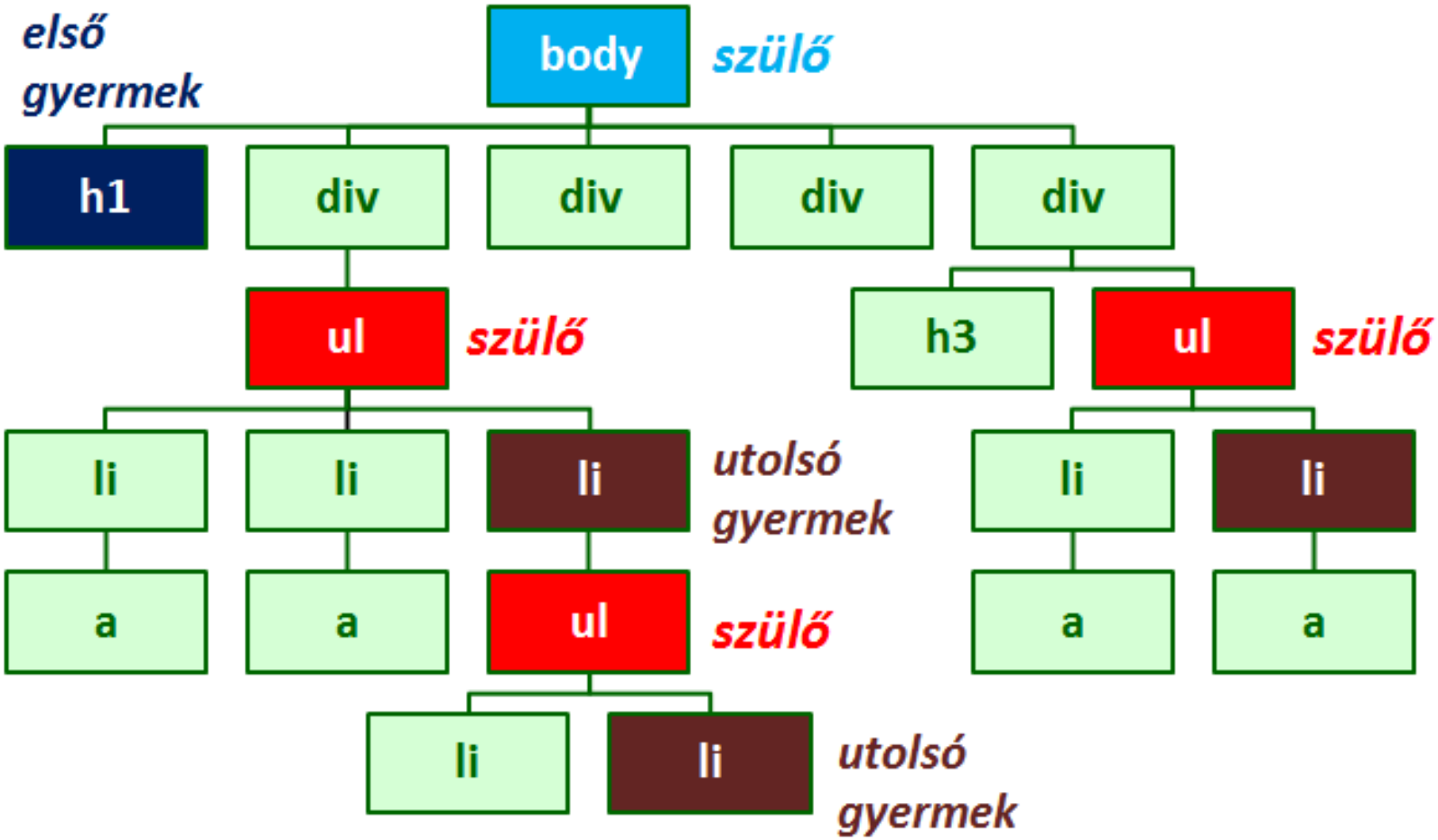
olyan elemet képvisel, amely valamilyen másik elemnek (szülő) az az első gyermeke – pl. táblázat első sora

(azaz egyezik az :nth-child(1) kijelöléssel)

`tagnév:last-child`

olyan elemet képvisel, amely valamilyen másik elemnek (szülő) az az utolsó gyermeke – pl. táblázat utolsó sora

példa dokumentumfával



IV/c) Strukturális ál-oszt. 5.

`tagnév:only-child`

olyan elemet képvisel, amely valamilyen másik elemnek (szülő) az az egyetlen gyermeke *(azaz a szülőnek nincsenek további leszármazottjai)*

`tagnév:nth-last-child(x)`

olyan elemet képvisel, amely valamilyen másik elemnek (szülő) hátulról a valahanyadik gyermeke

IV/c) Strukturális ál-oszt. 6.

`tagnév:nth-of-type(x)`

olyan elemet képvisel, amely valamilyen másik elemnek (szülő) a valahanyadik gyermeke az azonos típusú tagok között

pl. *** Ez az 1. bekezdés. ***
Ez az 1. blokkelem.
Ez a 2. blokkelem.
*** Ez a 2. bekezdés. ***
*** Ez a 3. bekezdés. *** → **p:nth-of-type(3)**
Ez a 3. blokkelem.

IV/c) Strukturális ál-oszt. 7.

pl. ha ***több kép*** is van a weboldalunkon közvetlenül ***ugyanabban az elemben gyermekként*** elhelyezve, akkor a képeket felváltva lehet formázni:

(páratlan "sorszámú" képek:)

```
img:nth-of-type(2n+1) { … }
```

(páros "sorszámú" képek:)

```
img:nth-of-type(2n) { … }
```

IV/c) Strukturális ál-oszt. 8.

`tagnév:first-of-type`

olyan elemet képvisel, amely valamilyen másik elemnek (szülő) az első gyermeke az azonos típusú tagok között

`tagnév:last-of-type`

olyan elemet képvisel, amely valamilyen másik elemnek (szülő) a legutolsó gyermeke az azonos típusú tagok között

IV/c) Strukturális ál-oszt. 9.

`tagnév:only-of-type`

olyan elemet képvisel, amely valamilyen másik elemnek (szülő) az egyetlen gyermeke a megadott típusú tagok között

`tagnév:nth-last-of-type(x)`

olyan elemet képvisel, amely valamilyen másik elemnek (szülő) hátulról a vala-hanyadik gyermeke az azonos típusú tagok között

V. Pszeudo-elem kijelölők

A **pszeudo-elemek** *(pseudo-elements)* segítségével **a HTML-tagok által meghatározott elemek bizonyos részeire hivatkozhatunk**, tehát a részelemek megcímzésére használjuk ezeket.

```
szelektor::pszeudo-elem
{ stílusdefiníció }
```

V/a) Első sor / első betű

Blokkszintű elemek első formázott sorára és/vagy betűire definiálható külön stílus

- ha minden bekezdés ***első sorát*** akarjuk formázni:

```
p::first-line { … }
```

- ha minden bekezdés ***első betűjét*** formázzuk:

```
p::first-letter { … }
```

Ezek is ál-osztályok, mert állapotuk a dokumentumban betöltött helyükről függ (első betű / első sor).

példa ::first-line használatára

```
p::first-line {
      color: #ff0000;
      font-variant: small-caps;  }
```

* A CSS-ben léteznek ún. látszólagos vagy pszeudo-osztályok és -elemek is, amelyek nem jelennek meg a dokumentum forrásában, sem a dokumentumfán. Segítségükkel speciális információk alapján hajthatunk végre formázásokat. *

A CSS-BEN LÉTEZNEK ÚN. LÁTSZÓLAGOS VAGY PSZEUDO-OSZTÁLYOK és -elemek is, amelyek nem jelennek meg a dokumentum forrásában, sem a dokumentumfán. Segítségükkel speciális információk alapján hajthatunk végre formázásokat.

példa ::first-letter használatára

```
p.bev::first-letter {
        color: #ff0000;
        font-size: 200%; }
```

* Bevezetés a pszeudo-osztályok használatába. *
* A pszeudo-osztályok megkülönböztetik az elemek típusait. A pszeudo-elemekkel a tag-ek által meghatározott elemek bizonyos részeire hivatkozhatunk, pl. egy bekezdés első betűjére. *

Bevezetés a pszeudo-osztályok használatába.

A pszeudo-osztályok megkülönböztetik az elemek típusait. A pszeudo-elemekkel a tag-ek által meghatározott elemek bizonyos részeire hivatkozhatunk, pl. egy bekezdés első betűjére.

V/b) Tartalom beillesztése

g/2) a forrásdokumentumban nem szereplő tartalom beillesztése egy elem elé vagy mögé

- ha minden bekezdés *elé* akarunk beilleszteni:

```
p::before { content: url(…); }
```

- ha minden bekezdés *után* akarunk tenni:

```
p::after { content: url(…); }
```

példa a ::before használatára

```
h1::before
      { content: url(konyv.gif); }
```

```
<h1>Főcím</h1>
<p>A ::before pszeudo-elem segítségével tartalmat illeszthetünk egy adott HTML-elem elé.</p>
<h2>Az első alcím</h2>
<p>Itt van az első alcímhez tartozó szöveg...</p>
<h2>A második alcím</h2>
<p>Itt van a második alcímhez tartozó szöveg...</p>
<h2>A harmadik alcím</h2>
<p>Itt van a harmadik alcímhez tartozó szöveg...</p>
```

példa a ::before használatára

Főcím

A ::before pszeudo-elem segítségével tartalmat illeszthetünk egy adott HTML-elem elé.

Az első alcím

Itt van az első alcímhez tartozó szöveg...

A második alcím

Itt van a második alcímhez tartozó szöveg...

A harmadik alcím

Itt van a harmadik alcímhez tartozó szöveg...

V/c) Kijelölt rész

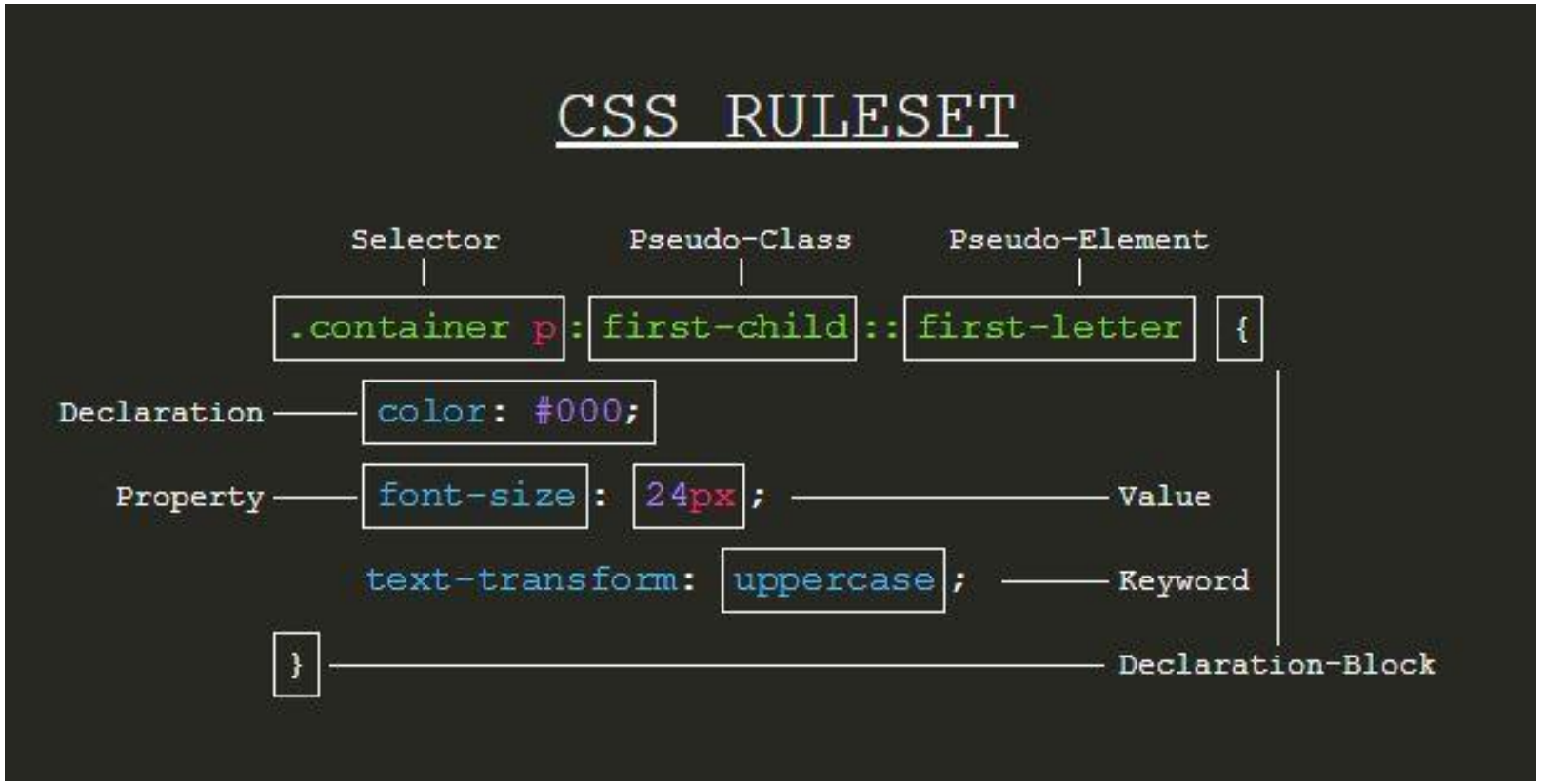
g/3) a forrásdokumentumban szereplő tartalom kijelölése esetén új formátum megjelenése

```
::selection { … }
```

A leggyakrabban alkalmazott CSS-tulajdonságok:

- *color – betűszín*
- *background – háttérszín*
- *cursor – a kurzor alakja*
- *outline – külső keret (nem a szegély !)*

ÖSSZEFOGLALÓ



Források

- CSS-alapok (weblabor.hu/cikkek/cssalapjai)
- w3schools.com
- HTML5 + CSS3 Szabványkövető statikus weboldalak szerkesztése
- Dr. Pál László: Web technológiák