Cascading Style Sheets (CSS)

Jeszenszky Péter
Debreceni Egyetem, Informatikai Kar
jeszenszky.peter@inf.unideb.hu

Utolsó módosítás: 2022. október 26.

Mi a CSS?

- Strukturált (például HTML és XML) dokumentumok megjelenítésének leírására szolgáló stíluslap nyelv.
  - Többféle eszközön történő megjelenítést támogat, mint például képernyők, nyomtatók, Braille eszközök.
- Szétválasztja a dokumentumok megjelenítési stílusát a dokumentumok tartalmától.
  - Ilyen módon egyszerűsíti a webszerkesztést és a webhelyek karbantartását.

Fejlesztés

- A W3C CSS Munkacsoportja fejleszti.
  - *CSS Working Group* https://www.w3.org/Style/CSS/members
  - *CSS Working Group Wiki* https://wiki.csswg.org/

Szintek

- A szó hagyományos értelmében a CSS-nek nincsenek verziói, hanem szintjei vannak:
  - *CSS Level 1*
  - *CSS Level 2*
  - *CSS Level 3*
- A CSS minden egyes szintje az előzőn alapul, annak definícióit finomítja és új lehetőségeket vezet be.
- Lásd: *Levels, snapshots, modules...* https://www.w3.org/Style/2011/CSS-process.en.html

CSS Level 1

- Specifikáció:
  - *Cascading Style Sheets, level 1* (túlhaladott W3C ajánlás) https://www.w3.org/TR/CSS1/
- Elavultnak tekintett.

CSS Level 2

- A CSS 2.1 specifikáció (egyetlen dokumentum) definiálja:
  - *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification* (W3C ajánlás, 2011. június 7.) https://www.w3.org/TR/CSS21/
- Javítása jelenleg fejlesztés alatt:
  - *Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification* (W3C munkaterv, 2016. április 12.) https://www.w3.org/TR/CSS22/

CSS Level 3 (1)

- Jelenleg is fejlesztés alatt áll.
- Moduláris felépítésű:
  - Modulokra van bontva, ahol minden egyes modul a CSS egy részét definiálja.
    - Minden egyes modul a CSS2.1 specifikáció lehetőségeit bővíti és/vagy részeit cseréli le.
- A moduloknak is szintjeik vannak.
  - Az 1. szintről indulnak az olyan modulok, melyeknek nincs megfelelője a *CSS Level 2*-ben.
  - A 3. szintről indulnak a *CSS Level 2* létező lehetőségeit frissítő modulok.

CSS Level 3 (2)

- A CSS modulok eltérő stabilitási szintűek.
- Az összes CSS specifikáció listája: http://www.w3.org/Style/CSS/current-work.en.html
- Néhány modul:
  - *Selectors Level 3* (W3C ajánlás, 2018. november 6.) https://www.w3.org/TR/selectors-3/
  - *CSS Values and Units Module Level 3* (előzetes W3C javaslatterv, 2019. június 6.) https://www.w3.org/TR/css-values-3/
  - *CSS Transforms Module Level 1* (előzetes W3C javaslatterv, 2019. február 14.) https://www.w3.org/TR/css-transforms-1/
  - *CSS Transforms Module Level 2* (W3C munkaterv, 2021. november 9.) https://www.w3.org/TR/css-transforms-2/
  - …

CSS Level 3 (3)

- *CSS Snapshot 2021* (W3C munkacsoport feljegyzés, 2021. december 31.) https://www.w3.org/TR/CSS/
  - A CSS jelenlegi állapotát alkotó specifikációkat gyűjti össze az implementálók számára.
  - Lásd a *Cascading Style Sheets (CSS) – The Official Definition* című részt. https://www.w3.org/TR/CSS/#css

CSS Level 4 és azon túl

- Nem létezik *CSS Level 4*.
- Önálló modulok elérhetnek a 4. vagy egy magasabb szintre, de a CSS nyelvnek már nincsenek a harmadikon túli szintjei.
- A *CSS Level 3* kifejezést csak a korábbi monolitikus verzióktól való megkülönböztetésre használják.

Böngésző támogatás

- A modern böngészőprogramok jórészt támogatják a CSS 2.1 szabványt.
- A CSS3 bizonyos lehetőségeit is támogatják.
  - Lásd:
    - https://caniuse.com/?search=CSS3
    - *The CSS3 Test* https://css3test.com/ https://github.com/LeaVerou/css3test

Szerkesztők (1)

- Szabad és nyílt forrású szoftverek:
  - *Visual Studio Code* (platform: Linux, macOS, Windows; licenc: *MIT License*)
    https://code.visualstudio.com/
    https://github.com/Microsoft/vscode
    - Lásd: https://code.visualstudio.com/docs/languages/css

Szerkesztők (2)

- Nem szabad szoftverek:
  - *WebStorm* (platform: Linux, macOS, Windows) https://www.jetbrains.com/webstorm/
    - Lásd: https://www.jetbrains.com/help/webstorm/style-sheets.html

További eszközök

- Kicsinyítő eszközök:
  - *cssnano* (programozási nyelv: CSS/JavaScript; licenc: *MIT License*) https://cssnano.co/ https://github.com/cssnano/cssnano
  - *CSSO* (programozási nyelv: CSS/JavaScript; licenc: *License MIT*) https://css.github.io/csso/ https://github.com/css/csso
  - *Visual Studio Code*:
    - *MinifyAll* (programozási nyelv: TypeScript; licenc: *GPLv3*) https://marketplace.visualstudio.com/items?itemName=josee9988.minifyall https://minifyall.jgracia.es/ https://github.com/Josee9988/MinifyAll

Online CSS homokozók

- *CodePen* https://codepen.io/
- *JSFiddle* https://jsfiddle.net/
- *Liveweave* https://liveweave.com/
- ...

Állomány jellemzők

- Állománynév végződés: `.css`
- IANA média típus: `text/css`

Karakterek

- Az Unicode karakterkészlet használata.

Vezérlősorozatok (1)

- Unicode karakterek megadásához használhatunk `\`*hhhhhh* formájú vezérlősorozatokat, ahol *hhhhhh* az Unicode karakter kódpontját ábrázoló legalább egy és legfeljebb 6 karakterből álló hexadecimális számjegysorozat.
  - Ha 6-nál kevesebb a számjegyek száma és a `[0-9a-fA-F]` karakterek valamelyike követi az utolsó számjegyet, akkor a vezérlősorozat végének jelzéséhez egy tetszőleges *whitespace* karaktert kell megadni.
    - Egy vezérlősorozatot követő *whitespace* karakter figyelmen kívül lesz hagyva.
  - Példa:
    - `\9`, `\09`, …, `\000009` a vízszintes tabulátor karaktert jelenti.
    - `\A9`, `\0A9`, …, `\0000A9` a *copyright* szimbólumot (©) jelenti.

Vezérlősorozatok (2)

- Speciális karakterek jelentésének elnyomásához használjuk a `'\'` karaktert.
  - Például szükséges akkor, ha kiválasztóban pont karaktert tartalmazó elemnevet kell megadni.
    - Például csak a given`\.`name kiválasztóra illeszkedik a given`.`name nevű elem, a given`.`name kiválasztóra nem!

Karakterkódolás

- A karakterkódolást az alábbiak határozzák meg:
  - A byte-sorrend jelző (*byte order mark*, BOM).
  - Az előbbi hiányában a `charset` paraméter értéke a `Content-Type` HTTP fejlécmezőben.
  - Az előbbi hiányában a stíluslap legelején a `@charset "`*kódolás*`";` at-szabály.
    - Példa: `@charset "UTF-16";`
  - Egyébként az UTF-8 karakterkódolás használata az alapértelmezés.
- Lásd:
  - *Declaring character encodings in CSS*
    https://www.w3.org/International/questions/qa-css-charset

Megjegyzések

- A `/*` és `*/` határólók között lehet megadni megjegyzéseket.
  - Példa: `/* Style sheet for index.html */`
- Tokeneken kívül bárhol megengedettek.
- Nem ágyazhatóak egymásba.

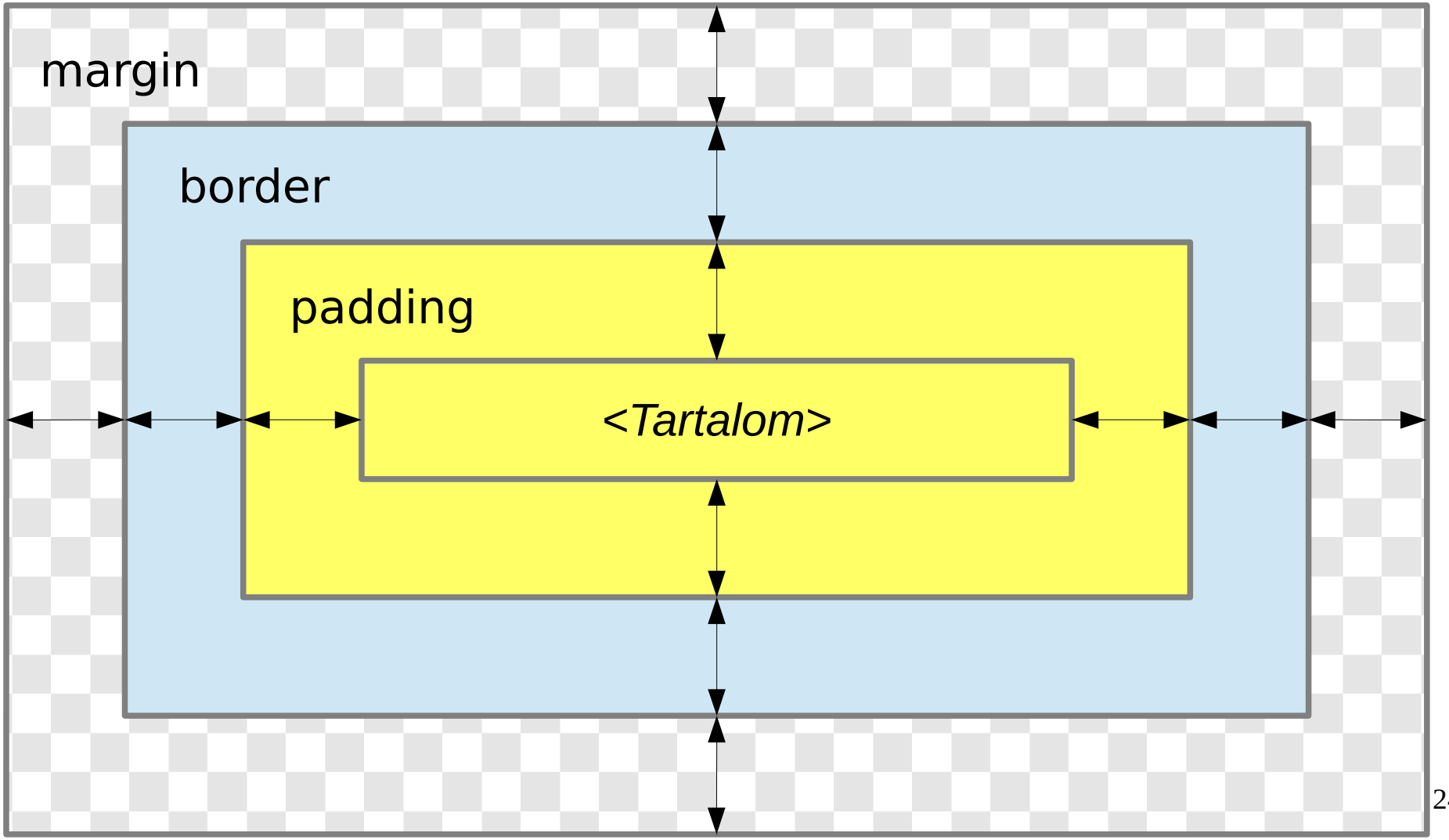
Dobozmodell (1)

- Vonatkozó specifikáció:
  - *CSS Display Module Level 3* (előzetes W3C javaslatterv, 2021. szeptember 3.) https://www.w3.org/TR/css-display-3/

Dobozmodell (2)

- A CSS egy fastruktúrájú dokumentumot kap, melyet egy rajzvásznon (például a képernyőn) jelenít meg egy olyan közbülső struktúrát, a **dobozfát** (***box tree***) előállítva, mely a megjelenített dokumentum formázási szerkezetét ábrázolja.
- Minden egyes doboz a fában a dokumentum egy megfelelő elemet (vagy pszeudo-elemét) ábrázolja térben és/vagy időben a rajzvásznon.
- A CSS minden egyes elemhez nulla vagy több dobozt generál az elem `display` tulajdonsága által meghatározott módon.
  - Egy elem általában egyetlen dobozt generál.

Dobozmodell (3)



Tulajdonságok (1)

- A CSS által definiált paraméterek, melyek révén a dokumentumok megjelenítése vezérelhető.
- A tulajdonságoknak neve és értéke van.
- Az összes tulajdonság listája:
  - *List of CSS properties* https://www.w3.org/Style/CSS/all-properties.en.html
    - Összesen 548 különböző tulajdonságnév.

Tulajdonságok (2)

- Összevont tulajdonság (*shorthand property*):
  - Olyan tulajdonság, mely több CSS tulajdonság értékének egyidejű beállítására szolgál.
  - Például a `margin` tulajdonság a `margin-top`, `margin-right`, `margin-bottom` és `margin-left` tulajdonságok értékét állítja be.

Deklarációs blokk

- `'{'` és `'}'` karakterek határolják, melyek között deklarációk egy listája kötelező.
  - A deklarációk *tulajdonságnév*`:`*érték* formájúak, ahol a tokenek előtt és után is megengedettek *whitespace* karakterek.
    - A deklarációban *tulajdonságnév* egy azonosító.
  - A deklarációkat `';'` karakterrel kell elválasztani.
    - Az utolsó deklaráció után nem kötelező a `';'` karakter.

Szabályhalmazok (stílus szabályok)

- Egy kiválasztóból (vagy `','` karakterekkel elválasztott kiválasztókból) és egy az(oka)t követő deklarációs blokkból állnak.

At-szabályok

- A stíluslap feldolgozását vezérlő speciális szabályok.
- `'@'` karakterrel kezdődnek, melyet egy azonosító követ, és `';'` karakterrel vagy egy deklarációs blokkal végződnek.
- Példa: `@charset`, `@import`, `@namespace`, `@media`
- Lásd: https://developer.mozilla.org/en-US/docs/Web/CSS/At-rule

Értékek (1)

- Vonatkozó specifikáció:
  - *CSS Values and Units Module Level 3* (előzetes W3C javaslatterv, 2019. június 6.) https://www.w3.org/TR/css3-values/
- Lásd még:
  - *CSS values and units* https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Values_and_units

Értékek (2)

- A tulajdonságok értékét komponens értékek alkotják:
  - Azonosítók (példa: `none`, `inherit`)
  - Sztringek (példa: `'Hello, world!'`)
  - URL-ek (példa: `url(images/item.png)`)
  - Számok (példa: `123`, `3.141593`)
  - Százalékok (példa: `150%`)
  - Dimenziók (példa: `10px`, `0.5em`, `45deg`)
  - Színek (példa: `fuchsia`, `#f0f`, `#ff00ff`, `rgb(255,0,255)`)
  - Funkcionális jelölések (példa: `attr(name)`, `calc(100% - 50px)`, `counter(chapter-number)`)
  - …

Értékek (3)

- Minden egyes tulajdonsághoz meghatározott az értékben használható komponens értékek típus, száma és sorrendje.
  - Példa:
    - `text-transform:` `none` | `capitalize` | `uppercase` | `lowercase` | `full-width`
    - `letter-spacing:` `normal` | <length>
    - `border-color:` <color>{1,4}
    - `border-top:` <line-width> || <line-style> || <color>

Azonosítók

- Kulcsszavak:
  - ASCII kisbetű-nagybetű érzéketlenek (ekvivalensek az `[a-z]` és `[A-Z]` karakterek).
- Egyéni azonosítók (*author-defined identifiers*):
  - Például számlálók neveként használatosak.
  - Kisbetű-nagybetű érzékenyek.
- Szintaxis:
  - Lásd: https://www.w3.org/TR/css-syntax-3/#ident-token-diagram

Sztringek

- `'''` vagy `'"'` karakterek között adhatóak meg.
- Nem tartalmazhatják a határoló karaktert.
  - Határólójelek levédéséhez használjuk a `'\'` karaktert.
    - Példa: `'It\'s So Cool',"\"Good morning, Frank,\" said Hal."`
- Újsor karakter kizárólag a \A vezérlősorozattal adható meg.

URL-ek

- Példa:
  - `list-style-image: url("http://eg.com/images/item.png")`
  - `list-style-image: url('http://eg.com/images/item.png')`
  - `list-style-image: url(http://eg.com/images/item.png)`
- Használható relatív hivatkozás, feloldásakor bázis-URL a stíluslap URL-je.
  - Példa: `list-style-image: url("images/item.png")`
- Bizonyos környezetekben (például `@import`) elhagyható az `url()` függvény és az URL megadható sztringként is.
  - Példa: `@import "default.css";`

Számok

- Kétféle szám használata:
  - **Egészek**: egy opcionális előjel karakterből és legalább egy decimális számjegy karakterből állnak.
    - Példa: `0`, `255`, `-1`, `+10946`
  - **Valós számok**: egy opcionális előjel karakterből, legalább egy decimális számjegy karakterből, egy opcionális decimális pont karakterből állnak, melyeket opcionálisan egy `'e'` vagy `'E'` karakter után egy egész szám követhet.
    - A decimális pont karakter jobb oldalán kötelező legalább egy számjegy karakter.
    - Speciális esetüket jelentik az egész számok.
    - Példa: `1.5`, `-100.0`, `+.25`, `1E-10`
- Tulajdonságok korlátozhatják az értékként használható számok tartományát.
  - Például kizárhatják a negatív számokat.

Százalékok (1)

- Egy százalék egy olyan egész vagy valós szám, melyet közvetlenül egy `'%'` karakter követ.
- Egy százalék mindig egy másik mennyiséghez viszonyított.
  - Minden olyan tulajdonság, melynek értékeként megengedett százalék, meghatározza, hogy a százalék mely mennyiségre utal.
  - Egy százalék lehet például az elem egy másik tulajdonságának értékéhez vagy egy felmenő elem egy adott tulajdonságának értékéhez viszonyított.

Százalékok (2)

- Példa:

```
sup { font-size: 75% }
nav { width: 40% }
```

Dimenziók

- Egy dimenzió egy olyan szám, melyet közvetlenül egy mértékegység követ.
  - A mértékegység egy ASCII kisbetű-nagybetű érzéketlen azonosító.
- Egy adott mennyiséget ábrázolnak, mint például hossz vagy szög.
- Hosszúság esetén a 0 értéknél elhagyható a mértékegység.

Hosszúságok (1)

- Kétféle mértékegység:
  - **Relatív**: egy másik hosszúsághoz képest határoz meg egy hosszúságot.
    - `em`, `ex`, `ch`, `rem`, `vw`, `vh`, `vmin`, `vmax`
  - **Abszolút**: valamilyen fizikai méreten alapul.
    - `cm`, `mm`, `Q`, `in`, `pc`, `pt`, `px`
- Példa: `0.5em`, `1.5cm`, `0px`, `0`, `-1in`
- Lásd: https://developer.mozilla.org/en-US/docs/Web/CSS/length

Hosszúságok (2)

- Relatív mértékegységek:
  - **`em`**: a `font-size` tulajdonság az elemhez tartozó számított értékét jelenti (a `font-size` tulajdonság esetén a `font-size` tulajdonság a szülő elemhez tartozó számított értékét).
  - **`ex`**: az elemhez tartozó betűkészlet *x-height* értékét jelenti (a `font-size` tulajdonság esetén a szülő elemhez tartozó *x-height* értéket).
    - Azért nevezik így, mert gyakran a betűkészlet *x* karakterének magasságával egyezik meg értéke.

Színek (1)

- Vonatkozó specifikáció:
  - *CSS Color Module Level 3* (W3C ajánlás, 2022. január 18.) https://www.w3.org/TR/css-color-3/

Színek (2)

- 16 szín kulcsszó: `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white`, `yellow`
- A modern böngészőprogramok támogatni szoktak számos további szín kulcsszót is.
  - Lásd: https://www.w3.org/TR/css-color-3/#svg-color

Színek (3)

- RGB színmodell:
  - Ekvivalens lehetőségek a halvány palaszürke (`lightslategrey`) szín megadására:
    - `#778899`
    - `#789` (minden egyes hexadecimális számjegy megkettőzése)
    - `rgb(119,136,153)`

Színek (4)

- RGB színmodell:
  - Az `rgb()` funkcionális jelölésnél használni lehet százalékokat is, ekkor 100% a 255 értéknek felel meg.
  - Ekvivalensek tehát például a következő színek is:
    - `#ffffff`
    - `#fff`
    - `rgb(255,255,255)`
    - `rgb(100%,100%,100%)`
    - `white`

Kiválasztók (1)

- Vonatkozó specifikáció:
  - *Selectors Level 3* (W3C ajánlás, 2018. november 6.) https://www.w3.org/TR/selectors-3/
- Mintaillesztésre szolgálnak.
- Meghatározzák, hogy egy szabály mely elemekre vonatkozik.

Kiválasztók (2)

- Kisbetű-nagybetű érzéketlenség az ASCII tartományban (ekvivalensek az `[a-z]` és `[A-Z]` karakterek) az olyan részek kivételével, melyek nem a CSS hatálya alá esnek (elemnevek, attribútumnevek és -értékek).
  - Az utóbbi esetben a kisbetű-nagybetű érzékenység a használt dokumentum nyelvtől függ.
    - Például a HTML kisbetű-nagybetű érzéketlen az elemnevek tekintetében, az XHTML és XML kisbetű-nagybetű érzékeny.

Kiválasztók (3)

- **Egyszerű kiválasztó**: egyetlen típus kiválasztó, általános kiválasztó, attribútum kiválasztó, osztály kiválasztó, ID-kiválasztó vagy pszeudo-osztály.
  - Példa:
    - `div`
    - `*`
    - `[rel=stylesheet]`
    - `.copyright`
    - `#logo`
    - `:lang(hu)`

Kiválasztók (4)

- **Kombinátor**: az alábbi karakterek valamelyike:
  - *whitespace*
  - '>'
  - '+'
  - '~'

Kiválasztók (5)

- **Egyszerű kiválasztók sorozata**:
  - Egymást követő egyszerű kiválasztók, melyek között nincsenek kombinátorok.
  - Típus kiválasztóval vagy általános kiválasztóval kezdődnek és nem tartalmaznak további típus kiválasztókat vagy általános kiválasztókat.
  - Példa:
    - `div`
    - `h2#status`
    - `link[rel=stylesheet][type=text/css]`
    - `p.copyright`
    - `*:lang(hu)`
    - `tr:nth-child(odd)`
    - `li:not(:last-child)`

Kiválasztók (6)

- **Kiválasztó**:
  - Egyszerű kiválasztók sorozatainak olyan sorozata, melyeket kombinátorok választanak el.
    - Legalább egy sorozat alkotja.
  - Pszeudo-elem az utolsó egyszerű kiválasztó sorozat végéhez adható hozzá, de legfeljebb csak egy!
  - Például:
    - `p`
    - `a img`
    - `h1 ~ table`
    - `div#main > h1`
    - `div > h1 + p`
    - `q::before`

Típus kiválasztó

- Egy CSS minősített név, a gyakorlatban tipikusan egy azonosító.
- A megfelelő nevű elemek illeszkednek rá.
- Példa:

```
p { color: red }
a { text-decoration: none }
```

Általános kiválasztó

- Általános kiválasztónak nevezzük a `*` formájú kiválasztót.
- Minden elem illeszkedik rá.
- Elhagyható olyan egyszerű kiválasztóból, mely további komponenseket is tartalmaz.
  - Így például az alábbi kiválasztók ekvivalensek:
    - `*#nav` és `#nav`
    - `*.important` és `.important`
    - `*[title]` és `[title]`
  - Nem ajánlott az elhagyás a jobban olvashatóság végett.
    - Például a `div *:first-child` kiválasztó jobban olvasható, mint a `div :first-child`, mert kevésbé keverhető össze a `div:first-child` kiválasztóval.

Attribútum kiválasztók (1)

- [*att*]
  - Az *att* attribútummal rendelkező elemek illeszkednek rá.
- [*att=érték*]
  - Olyan elemek illeszkednek rá, melyek *att* attribútumának értéke pontosan *érték.*
- [*att~=érték*]
  - Olyan elemek illeszkednek rá, melyek *att* attribútumának értéke *whitespace* karakterekkel elválasztott olyan szavak egy listája, melyek egyike megegyezik *érték*kel.

Attribútum kiválasztók (2)

- [*att*|=*érték*]
  - Olyan elemek illeszkednek rá, melyek *att* attribútumának értéke pontosan megegyezik *érték*kel, vagy pedig az *érték*- karaktersorozattal kezdődik.
  - Olyan attribútumokhoz használni, melyek értékeként nyelvváltozatok (például `en-US`, `en-GB`) jelenhetnek meg.
    - Azonban az `xml:lang` attribútumhoz és a HTML `lang` attribútumához a `:lang(`*C*`)` pszeudo-osztályt kell használni.

Attribútum kiválasztók (3)

- A CSS3 által bevezetett attribútum kiválasztók:
  - [*att*^=*érték*]
    - Olyan elemek illeszkednek rá, melyek *att* attribútumának értéke az *érték* előtaggal kezdődik.
  - [*att*$=*érték*]
    - Olyan elemek illeszkednek rá, melyek *att* attribútumának értéke az *érték* utótaggal végződik.
  - [*att**=*érték*]
    - Olyan elemek illeszkednek rá, melyek *att* attribútumának értékében legalább egyszer előfordul *érték.*

Attribútum kiválasztók (4)

- Az attribútum kiválasztókban *érték* azonosító vagy sztring.
  - Tehát például `[dir=rtl]`, `[dir='rtl']` és `[dir="rtl"]` ekvivalensek.

Attribútum kiválasztók (5)

- Példa:

```
style[type=italic] {
  font-style: italic;
}
style[type=bold] {
  font-weight: bold;
}
style[type=normal] {
  font-style: normal;
  font-weight: normal;
}

a[hreflang|=en] {
  text-decoration: line-through;
}
```

Osztály kiválasztó

- HTML dokumentumoknál a `[class~=`*érték*`]` attribútum kiválasztó helyett használható az ekvivalens `.`*érték* kiválasztó.
- Példa:

```
div.centered {
  margin-left: auto;
  margin-right: auto;
}
.important {
  color: red;
  text-decoration: underline;
}
```

ID-kiválasztó

- *#azonosító* formájú kiválasztó, az adott azonosítójú elem illeszkedik rá.
- Az azonosítót egy ID típusú attribútum kell, hogy szolgáltassa a dokumentumban.
  - A dokumentum nyelvtől függ, hogy mi ennek az attribútumnak a neve, például a HTML-ben `id`.
- Példa:

```
div#main {
  width: 50%;
  margin-left: auto;
  margin-right: auto;
}
#footer { text-align: center }
```

Pszeudo-osztályok (1)

- :*azonosító* vagy :*azonosító*(*érték*) formájú kiválasztók.
  - A pszeudo-osztályok neve kisbetű-nagybetű érzéketlen.
- Olyan kiválasztást tesznek lehetővé, mely a dokumentumon kívül információkon alapul vagy nem fejezhető ki a többi egyszerű kiválasztóval.
- Bizonyos pszeudo-osztályok egymást kölcsönösen kizáróak.

Pszeudo-osztályok (2)

- Dinamikus pszeudo-osztályok: olyan pszeudo-osztály, melyet egy elem megszerezhet vagy elveszíthet, miközben a felhasználó interakcióban van a dokumentummal.

Pszeudo-osztályok (3)

- Dinamikus pszeudo-osztályok:
  - **Link pszeudo-osztályok**:
    - `:link`: a felhasználó által még nem meglátogatott hiperhivatkozásokra vonatkozik.
    - `:visited`: a felhasználó által meglátogatott hiperhivatkozásokra vonatkozik.
  - **Felhasználói akció pszeudo-osztályok**:
    - `:hover`: a felhasználó által mutatóeszközzel kijelölt, de nem feltétlenül aktivált elemre vonatkozik.
    - `:active`: a felhasználó által mutatóeszközzel aktivált elemre vonatkozik.
      - Például az egérgomb lenyomása és felengedése között hatásos.
    - `:focus`: a fókuszt birtokló elemre vonatkozik.

Pszeudo-osztályok (4)

- Dinamikus pszeudo-osztályok:
  - Példa:

```
:link {
  border-width: medium;
  border-style: solid;
  text-decoration: none;
}

:visited {
    text-decoration: line-through
}

a:hover {
  text-decoration: overline underline;
  color: red;
}

a:active { font-weight: bolder }
```

Pszeudo-osztályok (5)

- A `:target` pszeudo-osztály:
  - Ha a dokumentum URI-ja tartalmaz erőforrásrész-azonosítót, akkor az erőforrásrész-azonosítónak megfelelő elem kiválasztása.
    - Ha például `index.html#copyright` a dokumentum URI-ja, akkor a `div:target` kiválasztó a `copyright` azonosítójú `div` elemet választja ki.

Pszeudo-osztályok (6)

- A `:lang(`*C*`)` pszeudo-osztály:
  - A *C* nyelvű szöveget tartalmazó elemek illeszkednek rá.
    - *C* egy CSS azonosító (nyelvkód).
  - Példa:
    - `:lang(en)`, `:lang(en-GB)`, `:lang(en-US)`
    - `:lang(hu)`
  - XML dokumentumokban a nyelvet az `xml:lang` attribútum határozza meg, HTML dokumentumokban a nyelv megadható a `lang` és az `xml:lang` attribútummal is.
    - A `lang` és `xml:lang` attribútum nem közvetlenül az elemen kell, hogy megjelenjen.

Pszeudo-osztályok (7)

- `A:lang(C)` pszeudo-osztály:
  - Példa: magyar nyelvű idézet esetén magyar idézőjelek használata

```
q:lang(hu) {
  quotes: "„" "”" "»" "«";
}
```

Pszeudo-osztályok (8)

- UI elem állapot pszeudo-osztályok:
  - **`:enabled`**: engedélyezett állapotú felhasználói felület elemek kiválasztása.
  - **`:disabled`**: letiltott állapotú felhasználói felület elemek kiválasztása.
  - **`:checked`**: bejelölt felhasználói felület elemek (például jelölőnégyzetek, rádiógombok) kiválasztása.

Pszeudo-osztályok (9)

- Szerkezeti pszeudo-osztályok:
  - Amikor megállapításra kerül, egy elem helye a testvéreinek listájában, akkor csak az elemeket kell a listában figyelembe venni.
    - Nem kell figyelembe venni például szöveget, megjegyzéseket és feldolgozási utasításokat sem.
  - A listában az elemek számozása egytől történik.

Pszeudo-osztályok (10)

- Szerkezeti pszeudo-osztályok:
  - **`:root`**: a dokumentum gyökérelemét választja ki.
  - **`:only-child`**: olyan elemeket választ ki, melyek szülőjének nincs más elemgyermeke.
  - **`:only-of-type`**: olyan elemeket választ ki, melyek szülőjének nincs más az elemmel megegyező kifejtett nevű elemgyermeke.
  - **`:empty`**: a szöveget és elemeket nem tartalmazó elemeket választja ki, melyek azonban tartalmazhatnak megjegyzéseket és feldolgozási utasításokat.

Pszeudo-osztályok (11)

- Szerkezeti pszeudo-osztályok (folytatás):
  - **`:nth-child(`*`an+b`*`)`**: ahol *a* és *b* egészek
    - Az olyan elemeket választja ki, melyeknek *an*+*b*-1 megelőző elemtestvére van valamely nemnegatív *n*-re.
  - **`:nth-last-child(`*`an+b`*`)`**: ahol *a* és *b* egészek
    - Az olyan elemeket választja ki, melyeknek *an*+*b*-1 következő elemtestvére van valamely nemnegatív *n*-re.
  - **`:nth-of-type(`*`an+b`*`)`**: ahol *a* és *b* egészek
    - Az olyan elemeket választja ki, melyeknek valamely nemnegatív *n*-re *an*+*b*-1 olyan megelőző elemtestvére van, melyek kifejtett neve megegyezik az elem kifejtett nevével.
  - **`:nth-last-of-type(`*`an+b`*`)`**: ahol *a* és *b* egészek
    - Az olyan elemeket választja ki, melyeknek valamely nemnegatív *n*-re *an*+*b*-1 olyan következő elemtestvére van, melyek kifejtett neve megegyezik az elem kifejtett nevével.

Pszeudo-osztályok (12)

- Szerkezeti pszeudo-osztályok (folytatás):
  - `:nth-child()`, `:nth-last-child()`, `:nth-of-type()` és `:nth-last-of-type()`:
    - Az *a*`n+0` argumentum megadható röviden *a*`n` módon.
    - A `0n+`*b* argumentum megadható röviden *b* módon.
    - Negatív *b* esetén *a*`n-`*b* módon kell megadni az argumentumot, nem megengedett *a*`n+-`*b*.
    - A `2n` argumentum megadható `even` módon, a `2n+1` argumentum `odd` módon.

Pszeudo-osztályok (13)

- Szerkezeti pszeudo-osztályok (folytatás):

Kiválasztó	Jelentés

`tr:nth-child(2n+0)`
`tr:nth-child(2n)`
`tr:nth-child(even)`
	A páros számú táblázatsorok

`tr:nth-child(2n+1)`
`tr:nth-child(2n-1)`
`tr:nth-child(odd)`
	A páratlan számú táblázatsorok

`tr:nth-child(5)`	Az ötödik táblázatsor
`tr:nth-child(5n+1)` `tr:nth-child(5n-4)`	Az 1, 6, 11, … számú táblázatsorok
`tr:nth-child(-n+5)`	Az első öt táblázatsor
`tr:nth-last-child(-n+5)`	Az utolsó öt táblázatsor

Pszeudo-osztályok (14)

- Szerkezeti pszeudo-osztályok (folytatás):
  - **`:first-child`**: azt jelenti, mint `:nth-child(1)`
  - **`:last-child`**: azt jelenti, mint `:nth-last-child(1)`
  - **`:first-of-type`**: azt jelenti, mint `:nth-of-type(1)`
  - **`:last-of-type`**: azt jelenti, mint `:nth-last-of-type(1)`

Pszeudo-osztályok (15)

- A negáció pszeudo-osztály:
  - `:not(`*X*`)` formában adható meg, ahol az argumentum egy olyan egyszerű kiválasztó, melyben nem megengedett a negáció pszeudo-osztály.
  - Azok az elemek illeszkednek rá, melyek nem illeszkednek az argumentumra.
  - Példa:
    - `:not(.important)`
    - `:not([title])`
    - `li:not(:last-child)`
    - `a:not(:hover)`
  - A specifikusság meghatározásakor csak az argumentumot kell figyelembe venni!
    - Például a `h1:not(h2)` kiválasztó gyakorlatilag ekvivalens a `h1` kiválasztóval, de nagyobb a specifikussága: (*a* = 0, *b* = 0, *c* = 2).

Pszeudo-elemek (1)

- Lehetővé teszik a dokumentumok olyan részeinek kiválasztását, melyek más módon nem hozzáférhetők.
- Minden kiválasztókban legfeljebb egy pszeudo-elem megengedett.
- Az alábbi pszeudo-elemek állnak rendelkezésre:
  - `::first-line`
  - `::first-letter`
  - `::before`, `::after`
- A pszeudo-osztályoktól való megkülönböztetés miatt `':'` helyett `'::'` karaktereket használ a pszeudo-elemekhez a CSS3.
  - Kompatibilitási okokból használható `':'` is.

Pszeudo-elemek (2)

- `::first-line`: egy elem első formázott sorát ábrázolja.
  - Példa:

```
p::first-line {
  text-decoration: underline
}
```

Pszeudo-elemek (3)

- `::first-letter`: egy elem első betű vagy számjegy karakterét ábrázolja, ha azt nem előzi meg más tartalom, például kép.
  - Vonatkozik az első betű vagy számjegy karaktert megelőző vagy követő írásjelekre is.
  - Példa:

```
p::first-letter {
  font-size: 2em;
  font-weight: bold;
}
```

Pszeudo-elemek (4)

- `::before` és `::after`: lehetővé teszik egy elem tartalmához generált tartalom hozzáadását.
  - Példa:

```
div.proof::before {
  content: "Proof: ";
  font-weight: bold;
}

div.proof::after {
   content: "\220E" /* End of proof */
}
```

Kombinátorok (1)

- Leszármazott kombinátor:
  - Egyszerű kiválasztók két sorozatát elválasztó *whitespace*.
  - Ha *P* és *Q* egyszerű kiválasztók két sorozata, akkor a *P Q* kiválasztóra a *Q*-ra illeszkedő olyan elemek illeszkednek, melyek a *P*-re illeszkedő elemek leszármazottai.
  - Példa:

```
thead th { background-color: darkgrey }
```

Kombinátorok (2)

- Gyermek kombinátor:
  - Egyszerű kiválasztók két sorozatát elválasztó `'>'` karakter.
  - Ha *P* és *Q* egyszerű kiválasztók két sorozata, akkor a *P* > *Q* kiválasztóra a *Q*-ra illeszkedő olyan elemek illeszkednek, melyek a *P*-re illeszkedő elemek gyermekei.
  - Példa:

```
nav > div { display: inline }

p > img:only-child { margin-left: 0 }
```

Kombinátorok (3)

- Szomszéd testvér kombinátor:
  - Egyszerű kiválasztók két sorozatát elválasztó `'+'` karakter.
  - Ha *P* és *Q* egyszerű kiválasztók két sorozata, akkor a *P* + *Q* kiválasztóra a *Q*-ra illeszkedő olyan elemek illeszkednek, melyek a *P*-re illeszkedő elemet követnek közvetlenül a dokumentumban.
    - Az illeszkedő elemeknek ugyanaz kell, hogy legyen a szülője.
    - Közöttük megengedettek olyan konstrukciók, melyek nem elemek, például szöveg és megjegyzések, ezek figyelmen kívül hagyása.
  - Példa:

```
h1 + p { text-indent: 0 }
```

Kombinátorok (4)

- Általános testvér kombinátor (CSS3):
  - Egyszerű kiválasztók két sorozatát elválasztó `'~'` karakter.
  - Ha *P* és *Q* egyszerű kiválasztók két sorozata, akkor a *P* ~ *Q* kiválasztóra a *Q*-ra illeszkedő olyan elemek illeszkednek, melyek a *P*-re illeszkedő elemet követnek (nem feltétlenül) közvetlenül a dokumentumban.
    - Az illeszkedő elemeknek ugyanaz kell, hogy legyen a szülője.
  - Példa:

```
img ~ span { color: red }
```

Szabályok összevonása (1)

- Összevonható több olyan stílus szabály, melyekhez azonos deklarációs blokk tartozik.
  - Az összevont szabályban egy olyan kiválasztót kell használni, melyben az egyes szabályok kiválasztóit `','` karakterekkel választjuk el egymástól.

Szabályok összevonása (2)

- Például ekvivalens a felső három szabály a negyedikkel:

```
h1 + p { text-indent: 0 }
h2 + p { text-indent: 0 }
h3 + p { text-indent: 0 }

h1 + p, h2 + p, h3 + p {
   text-indent: 0
}
```

Selectors Level 4 (1)

- Specifikáció:
  - *Selectors Level 4* (W3C munkaterv, 2022. május 7.) https://www.w3.org/TR/selectors-4/
- Változások a *Level 3* szinthez képest: https://www.w3.org/TR/selectors-4/#changes-level-3
- Új lehetőségek:
  - `:has()` pszeudo-osztály
  - `:is()` pszeudo-osztály
  - `:not()`: megadható argumentumként kiválasztó lista
  - ...

Selectors Level 4 (2)

- `:is()` pszeudo-osztály: `:is(`*X*`)` formában adható meg, ahol az argumentum egy kiválasztó lista.
  - *Matches-any pseudo-class* néven is ismert.
  - Minden olyan elem illeszkedik rá, mely illeszkedik a lista argumentum valamely kiválasztójára.
  - Hosszú kiválasztó listák tömörebb alakban való ábrázolásához hasznos.
  - Az `:is()` kiválasztó specifikussága az argumentum lista legnagyobb specifikusságú kiválasztójának specifikussága.
  - Böngésző támogatás: https://caniuse.com/css-matches-pseudo

Selectors Level 4 (3)

- `:is()` pszeudo-osztály: ekvivalens például az alábbi két stíluslap szabály:

```
div:is(.note, .warning, .hint)::before {
  /* ... */
}

div.note::before,
div.warning::before,
div.hint::before {
  /* ... */
}
```

CSS kiválasztók egyéb felhasználásai

- Információkinyerés weboldalakból (*web scraping*)
  - Szabad és nyílt forrású szoftverek:
    - *Scrapy* (programozási nyelv: Python; licenc: *New BSD License*) https://scrapy.org/ https://github.com/scrapy/scrapy
    - *Parsel* (programozási nyelv: Python; licenc: *New BSD License*) https://parsel.readthedocs.io/ https://github.com/scrapy/parsel

Specifikusság (1)

- Kiválasztókhoz és deklarációkhoz **specifikusság** meghatározása.
- A specifikusság egy háromelemű (*a*, *b*, *c*) vektor, ahol *a*, *b* és *c* nemnegatív egészek.
- A vektorok rendezése lexikografikusan történik.

Specifikusság (2)

- Kiválasztó specifikusságának meghatározása: a specifikusság egy (*a*, *b*, *c*) vektor, ahol:
  - *a* a kiválasztóban előforduló ID-kiválasztók száma.
  - *b* a kiválasztóban előforduló attribútum kiválasztók és pszeudo-osztályok száma.
    - A negáció pszeudo-osztályt figyelmen kívül kell hagyni, azonban az argumentumát nem!
  - *c* a kiválasztóban előforduló típus kiválasztók és pszeudo-elemek száma.

Specifikusság (3)

- Deklarációk specifikussága:
  - Egy deklaráció specifikussága megegyezik a tartalmazó szabály kiválasztójának specifikusságával.
  - Szabályhoz nem tartozó (HTML-ben a `style` attribútum értékeként adott) deklaráció specifikussága nagyobb minden kiválasztóénál.

Példák a specifikusság meghatározására (1)

- A `*` kiválasztó specifikussága (*a* = 0, *b* = 0, *c* = 0).
  - Mivel nem tartalmaz sem ID-kiválasztót, sem attribútum kiválasztót, sem pszeudo-osztályt, sem típus kiválasztót, sem pszeudo-elemet.
- A `div` kiválasztó specifikussága (*a* = 0, *b* = 0, *c* = 1).
- A `p::first-letter`, az `a img` kiválasztó és a `h1 + p` kiválasztó specifikussága (*a* = 0, *b* = 0, *c* = 2).

Példák a specifikusság meghatározására (2)

- A `div[class=nav]` és az ekvivalens `div.nav` kiválasztó specifikussága (*a* = 0, *b* = 1, *c* = 1).
- A `#main *:lang(en)` kiválasztó specifikussága (*a* = 1, *b* = 1, *c* = 0).

Példák a specifikusság meghatározására (3)

- A példákban szereplő kiválasztók specifikussága növekvő sorrendben:

Kiválasztó	Specifikusság
`*`	($a = 0, b = 0, c = 0$)
`div`	($a = 0, b = 0, c = 1$)
`p::first-letter`	($a = 0, b = 0, c = 2$)
`a img`	
`h1 + p`	
`div[class=nav]`	($a = 0, b = 1, c = 1$)
`div.nav`	
`#main *:lang(en)`	($a = 1, b = 1, c = 0$)

Különböző eredetű stíluslapok

- Különböző eredetű stíluslapok állhatnak rendelkezésre dokumentumok megjelenítéséhez.
- Egy stíluslap eredete szerint lehet:
  - **A felhasználói ágenstől (böngészőtől) származó stíluslap**
  - **A felhasználótól származó stíluslap**
  - **A dokumentum szerzőjétől származó stíluslap**
- A fenti felsorolásban növekvő „erősorrendben” tüntettük fel a stíluslapokat.

A felhasználói ágenstől származó stíluslap (1)

- A felhasználói ágensek biztosítanak alapértelmezett stíluslapot.
  - Például olyan stílus szabály tartalmazása, mely az `em` HTML elem megjelenítéséhez kurzív betűtípust ír elő.

A felhasználói ágenstől származó stíluslap (2)

- **Firefox**: az alábbi módon férhetünk hozzá az alapértelmezett stíluslaphoz:
  - Írjuk be a böngésző címsorába a következő URI-t: `resource://gre-resources/`
  - Egy könyvtárat kapunk, melyben a `html.css` állomány az alapértelmezett stíluslap.
  - Lásd: https://searchfox.org/mozilla-central/source/layout/style/res/html.css
- **Chromium**, **Google Chrome**: az alapértelmezett stíluslap HTML dokumentumokhoz:
  - https://chromium.googlesource.com/chromium/blink/+/master/Source/core/css/html.css

A felhasználótól származó stíluslap

- A felhasználó megadhat saját stíluslapot adott dokumentum megjelenítéséhez.
  - Ez a fogyatékkal élő felhasználók számára fontos.
    - Kapcsolódó fogalom: webes akadálymentesítés (*web accessibility*) https://www.w3.org/WAI/
  - Erre szolgál például a Chromium, Google Chrome, Firefox és Opera böngészőkhöz rendelkezésre álló *Stylish* kiterjesztés. https://userstyles.org/

A szerzőtől származó stíluslap (1)

- (X)HTML esetén a `link` fejléc elemmel adható meg a dokumentumhoz külső stíluslap.
  - Példa:
    - `<link rel="stylesheet" href="style.css"/>`
- (X)HTML esetén használhatjuk a `style` fejléc elemet is, mellyel közvetlenül ágyazhatunk be stílus szabályokat a dokumentumba:
  - Példa:
    - ```
      <style>
        h1, h2, h3, h4, h5, h6 { font-variant: small-caps }
      </style>
      ```
      ```

A szerzőtől származó stíluslap (2)

- XML esetén az `xml-stylesheet` feldolgozási utasítással adható meg a dokumentumhoz külső stíluslap.
  - A feldolgozási utasítás a dokumentum gyökéreleme előtt kell, hogy szerepeljen.
  - Példa:
    - `<?xml-stylesheet type="text/css" href="style.css"?>`
  - Lásd: *Associating Style Sheets with XML documents 1.0 (Second Edition)* (W3C ajánlás, 2010. október 28.) https://www.w3.org/TR/xml-stylesheet/

Stíluslapok importálása

- Az `@import` at-szabály szabályok más stíluslapokból történő importálására szolgál.
- A stíluslap URL-je megadható az `url()` függvénnyel vagy sztringként is.
- Ha egy `@import` at-szabály egy érvényes stíluslapra hivatkozik, akkor a felhasználói ágensek úgy kell, hogy kezeljék annak tartalmát, mintha az az at-szabály helyén szerepelne.
- Az `@import` at-szabályok a `@charset` at-szabály kivételével meg kell, hogy előzzék az összes többi at-szabályt és stílus szabályt.
- Példa:

```
@import url(https://fonts.googleapis.com/css?family=Tangerine);
@import "default.css";
```

„Fontos” deklarációk (1)

- Egy deklarációt követheti a `'!'` token és az `important` kulcsszó.
  - Példa:
    - `color: red !important`
- Egy ilyen deklaráció felülír bármely más közönséges deklarációt.
- Lásd:
  - *Specificity – The `!important` exception*
    https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity#the_!important_exception

„Fontos” deklarációk (2)

- Alapértelmezésben a szerzői stíluslap szabályok nagyobb precedenciával bírnak, mint a felhasználói stíluslap szabályok.
  - A szerzői és a felhasználói stíluslapok is tartalmazhatnak `!important` deklarációkat, ilyenkor ezek precedenciája megfordul: a felhasználói `!important` szabályok felülírják a szerzői `!important` szabályokat.

„Fontos” deklarációk (3)

- Példa:

```
<html lang="en">
  <head>
    <title>!important</title>
    <meta charset="UTF-8">
    <style>
      p {
        background-color: cornsilk;
        color: green !important;
      }
    </style>
  </head>
  <body>
    <p style="background-color: aliceblue; color: red">
      Hello, World!
    </p>
  </body>
</html>
```

Tulajdonság értékének meghatározása

- Egy CSS tulajdonság végső értékének meghatározása több lépésben történik:
  - **Kaszkádolt érték** (***cascaded value***): a kaszkád eredménye.
  - **Meghatározott érték** (***specified value***): az alapértelmezés eredménye.
  - **Számított érték** (***computed value***): egy relatív érték abszolúttá történő átalakításának eredménye.
  - **Használt érték** (***used value***): a dokumentum formázásának eredménye.
  - **Tényleges érték** (***actual value***): a használt érték a megjelenítési környezetnek megfelelő értékké alakításának eredménye.

Kaszkádolás, öröklés és kezdőértékadás

- **Kaszkádolás** (***cascading***): konfliktusfeloldási mechanizmus arra az esetre, amikor egy elem/tulajdonság kombinációhoz különböző deklarációk állítanak be értéket.
- **Öröklés** (***inheritance***), **kezdőértékadás** (***initialization***): abban az esetben alkalmazásra kerülő mechanizmusok, amikor egy elem/tulajdonság kombinációhoz egy deklaráció sem állít be értéket.
- Vonatkozó specifikáció:
  - *CSS Cascading and Inheritance Level 3* (W3C ajánlás, 2021. február 11.) https://www.w3.org/TR/css-cascade-3/

Kaszkád (1)

- Több különböző deklaráció szolgáltathatja egy tulajdonság értékét egy elemhez.
  - Ezek a deklarációk különböző eredetűek is lehetnek.
- A kaszkád az a folyamat, melynek során meghatározásra kerül, hogy a vonatkozó deklarációk közül melyik határozza meg egy adott elem egy adott tulajdonságának értékét.

Kaszkád (2)

- A kaszkád az alábbi módon határozza meg egy adott elemhez egy adott tulajdonság kaszkádolt értékét:
  - Meg kell határozni azokat a deklarációkat, melyek az adott elemhez az adott tulajdonság értékét szolgáltatják.
    - Ne feledjük, hogy a deklarációkat stílus szabályok tartalmazzák vagy pedig a `style` HTML attribútum.
  - Rendezzük a vonatkozó deklarációkat az eredetük szerint csökkenő „erősorrendbe”.
  - Az azonos eredetű deklarációkat rendezzük specifikusság szerint csökkenő sorrendbe.
    - Azonos specifikusság esetén döntsön az előfordulási sorrend, két azonos specifikusságú deklaráció esetén a későbbi az „erősebb”.
  - A tulajdonság értékét a fenti sorrendben első deklaráció szolgáltatja.

Szabályok sorrendje (1)

- Lényeges lehet a szabályok sorrendje.
  - Akkor számíthat a sorrend, ha egy elemre egynél több azonos specifikusságú stílus szabály vonatkozik.
  - Ezek közül mindig a sorrendben utolsó a „legerősebb”.

Szabályok sorrendje (2)

- Például az alábbi sorrend mellett az első két szabály soha nem lesz alkalmazva!
  - ```
    a:active { color: red }     /* (a = 0, b = 1, c = 1) */
    a:hover { color: green }    /* (a = 0, b = 1, c = 1) */
    a:visited { color: black } /* (a = 0, b = 1, c = 1) */
    a:link { color: blue }      /* (a = 0, b = 1, c = 1) */
    ```
- Ez a fenti példában szereplő szabályok helyes sorrendje (az első két szabály felcserélhető):
  - ```
    a:visited { color: black }
    a:link { color: blue }
    a:hover { color: green }
    a:active { color: red }
    ```

Meghatározott érték

- Egy tulajdonság meghatározott értéke az az érték, melyet a stíluslap szerzője az elemhez szán.
- Ha a kaszkád egy értéket ad, akkor az a meghatározott érték.
  - Egyébként az alapértelmezés szolgáltatja a meghatározott értéket.

Alapértelmezés

- Amikor a kaszkád nem eredményez egy értéket, akkor a meghatározott értéket más módon kell megállapítani.
  - Az öröklött tulajdonságok a szülő elemtől kapnak alapértelmezett értéket az öröklés révén.
  - Az összes többi tulajdonság a kezdőértékét veszi fel a kezdőértékadás révén.
- A `inherit` és `initial` kulcsszavak révén kérhető explicit módon öröklés vagy kezdőértékadás.

Számított érték (1)

- A számított érték meghatározásakor egy relatív érték általában egy abszolút értékké kerül átalakításra.
- A számított érték lehet azonban százalék!
  - Számított értékül egy százalékot eredményez egy olyan százalékértékek, mely egy olyan mennyiséghez viszonyított, melynek meghatározása az elrendezéstől függ.
    - Ezek a használt érték meghatározásakor kerülnek átalakításra abszolút értékekké.
- A számított érték kerül továbbadásra az öröklésnél a szülőtől a gyermekekhez.

Számított érték (2)

- Példa:
  - A `section` elem `font-size` tulajdonságának meghatározott és számított értéke `12px`.
  - A `h1` elem `font-size` tulajdonságának meghatározott értéke `1.5em`, számított értéke `1.5 × 12px = 18px`.

```
section { font-size: 12px }
h1 { font-size: 1.5em }

<section>
  <h1>Introduction</h1>
  ...
</section>
```

Használt érték

- A számított érték meghatározásához nem kell tekintettel lenni a dokumentum elrendezésére.
- Vannak azonban olyan tulajdonságok (például `height`, `width`), melyek értéke függhet az elrendezéstől.
  - Ezek esetén az elrendezést figyelembe véve kerül meghatározásra a használt érték a számított értékből.
  - A többi tulajdonság esetén a használt érték megegyezik a számított értékkel.

Tényleges érték

- A használt érték a lokális környezetnek megfelelő átalakításával nyerhető a tényleges érték.
  - Például a `font-size` tulajdonság értékét alkalmas módon kell megváltoztatni, ha a szükséges betűkészlet nem áll rendelkezésre a megfelelő méretben.

Öröklés (1)

- Az öröklés a tulajdonságértékek továbbadását jelenti a szülő elemektől a gyermek elemekhez.
- Bizonyos tulajdonságok öröklöttek, ami azt jelenti, hogy az értékük öröklés révén kerül meghatározásra, feltéve, hogy a kaszkád nem eredményez egy értéket.
- A specifikáció minden egyes tulajdonsághoz meghatározza, hogy öröklött-e.
  - Öröklött például a `font-family`, `font-style`, `font-variant`, `font-weight`, `font-size` és `font` tulajdonság.
  - Nem öröklöttek például a `margin-top`, `margin-bottom`, `margin-right`, `margin-left` és `margin` tulajdonságok, sem a `width` tulajdonság.
- Egy tulajdonság kaszkádolt értékeként az `inherit` kulcsszó az öröklést kényszeríti ki.

Öröklés (2)

- Példa:
  - Ha az alábbi `em` HTML elemre nem vonatkozik a `color` tulajdonság értékét szolgáltató egyetlen stílus szabály sem, akkor a `span` elemtől örökli a `color` tulajdonság értékét (`red`).

```
<span style="color: red">
  Az <em>erő</em> legyen veled!
</span>
```

Öröklés (3)

- Példa:
  - Az alábbi vonatkozó stílus szabály esetén minden hiperhivatkozás a szülő elemtől örökli a `color` tulajdonság értékét. Így például a `div` elemben lévő a elem `color` tulajdonságának értéke `green`.

```
a:visited, a:link {
  color: inherit
}

<p style="color: green">
  Click here: <a href="https://www.w3.org/">W3C</a>
</p>
```

Kezdőérték

- Minden tulajdonságnak van egy kezdőértéke, melyet a CSS specifikációk határoznak meg. A kezdőérték előírható úgy, hogy az felhasználói ágenstől függ.
  - Például a `background-color` tulajdonság kezdőértéke a `transparent` kulcsszó.
    - Lásd: https://www.w3.org/TR/css-backgrounds-3/#the-background-color
  - Például a `color` tulajdonság kezdőértéke a felhasználói ágenstől függ.
    - Lásd: https://www.w3.org/TR/css-color-3/#foreground
- Ha a kaszkád nem eredményez egy értéket és a tulajdonság nem öröklött, akkor a tulajdonság meghatározott értéke a kezdőérték.
- Egy tulajdonság kaszkádolt értékeként az `initial` kulcsszó azt eredményezi, hogy a kezdőérték lesz a meghatározott érték.

Feladat (1)

- Tekintsük az alábbi szabályokat, melyeket feltevés szerint a szerző stíluslapja tartalmaz egy XHTML dokumentumhoz:

```
:lang(hu) { color: red }
div > p { color: blue }
#main > p.foo { color: yellow }
#main :lang(hu) { color: green }
div:lang(hu) p.foo:first-child { color: black }
```

- Milyen színű lesz az alábbi p elemben tartalmazott szöveg?

```
<div id="main" lang="hu">
  <p class="foo">Milyen színű ez a szöveg?</p>
</div>
```

Feladat (2)

- Tekintsük az alábbi szabályokat, melyeket feltevés szerint a szerző stíluslapja tartalmaz egy XHTML dokumentumhoz:

```
.movies > ul > li:not(:first-child) { color: navy }
h1 + ul li:first-child { color: gold }
#ot li + :last-child { color: tomato }
ul > :nth-child(2n + 1) { color: purple }
*:not([class]) + ul:first-child > li:first-child {
  color: silver
}
```

- Milyen színű lesz az egyes `li` elemekben tartalmazott szöveg?

```
<section id="ot" class="movies">
  <h1>The Original Trilogy</h1>
  <ul>
    <li>A New Hope</li>
    <li>The Empire Strikes Back</li>
    <li>Return of the Jedi</li>
  </ul>
</section>
```

Gyártói előtag (1)

- A CSS2.1 specifikáció egy előtagos szintaxist tart fenn a CSS gyártói és kísérleti kiterjesztéseihez.
  - A CSS egy lehetősége egy **gyártói kiterjesztés**, ha azt egy zárt környezetben való használatra szánják és csupán egyetlen gyártó felhasználói ágense(i) számára elérhető.
  - A CSS egy lehetőségét **instabil**nak tekintjük addig, míg a specifikációja el nem éri az előzetes javaslatterv (CR – *Candidate Recommendation*) szintet.
- Lásd: *CSS Snapshot 2021 – Implementations of Unstable and Proprietary Features* https://www.w3.org/TR/CSS/#future-proofing

Gyártói előtag (2)

- A `'-'` vagy `'_'` karakterrel kezdődő kulcsszavak és tulajdonságnevek gyártóspecifikus kiterjesztések számára vannak fenntartva.
  - Garantált, hogy a CSS jelenlegi és jövőbeli szintjei sem használják ezen karaktereket tulajdonságnév vagy kulcsszó elején.
- Lásd: *CSS 2.1 Specification – Syntax and basic data types – Vendor-specific extensions* https://www.w3.org/TR/CSS21/syndata.html#vendor-keywords

Gyártói előtag (3)

- Miután egy instabil lehetőség stabilizálódott – azaz a specifikációja eléri az előzetes javaslatterv szintet –, az implementációkból ajánlott eltávolítani a gyártói előtagos szintaxis támogatását.

Gyártói előtag (4)

- A nevezetes gyártói előtagok közé tartoznak az alábbiak:
  - **Mozilla** (Firefox): `-moz-`
  - **Microsoft** (Internet Explorer, Microsoft Edge): `-ms-`
  - **Apple** (Safari): `-webkit-`

Gyártói előtag (5)

- A Chromium nem használ gyártói előtagokat.
  - Helyettük egy böngésző beállítás szolgál (`chrome://flags/#enable-experimental-web-platform-features`) a kísérleti lehetőségek engedélyezéséhez/letiltásához.
  - Lásd: https://www.chromium.org/blink/developer-faq/#will-we-see-a-chrome-vendor-prefix-now

Gyártói előtag (6)

- Gyártói CSS kiterjesztések:
  - *Microsoft CSS extensions*
    https://developer.mozilla.org/en-US/docs/Web/CSS/Microsoft_extensions
  - *Mozilla CSS extensions*
    https://developer.mozilla.org/en-US/docs/Web/CSS/Mozilla_Extensions
  - *WebKit CSS extensions*
    https://developer.mozilla.org/en-US/docs/Web/CSS/Webkit_Extensions

Gyártói előtag (7)

- Példa gyártói kiterjesztésre: `::-moz-list-number`
  https://developer.mozilla.org/en-US/docs/Web/CSS/:-moz-list-number
  - Példa a használatra:

```
li::-moz-list-number {
  font-style: italic;
  font-weight: bold;
}
```

Gyártói előtag (8)

- Példa instabil lehetőségre: `fit-content`
  https://developer.mozilla.org/en-US/docs/Web/CSS/fit-content
  - Vonatkozó CSS specifikáció: *CSS Box Sizing Module Level 4* (W3C munkaterv, 2021. május 20.) https://www.w3.org/TR/css-sizing-4/
  - Böngésző támogatás: https://caniuse.com/mdn-css_properties_width_fit-content

```
.fit-content {
  width: -moz-fit-content;
  width: fit-content;
}
```

CSS Houdini (1)

- A Houdini egy alacsony szintű API-készlet, mely felfedi a CSS motor részeit, lehetővé téve a fejlesztők számára a CSS kiterjesztését a böngésző renderelő motorjának stilizálási és elrendezési folyamatába való beavatkozással.
  - Lásd: https://developer.mozilla.org/en-US/docs/Web/Houdini

CSS Houdini (2)

- Specifikációk:
  - *CSS-TAG Houdini Editor Drafts* https://drafts.css-houdini.org/
- Állapot és támogatás:
  - *Is Houdini ready yet?* https://ishoudinireadyyet.com/
  - https://caniuse.com/?search=houdini

CSS Houdini (3)

- További hasznos linkek:
  - *CSS Houdini Wiki* https://github.com/w3c/css-houdini-drafts/wiki
  - *CSS Houdini Experiments* https://css-houdini.rocks/ https://github.com/iamvdo/css-houdini.rocks
  - *Houdini.how* https://houdini.how/
  - *Houdini Spellbook* https://houdini.glitch.me/

További hasznos linkek (1)

- *MDN Web Docs – CSS* https://developer.mozilla.org/en-US/docs/Web/CSS
  - *CSS Reference* https://developer.mozilla.org/en-US/docs/Web/CSS/Reference
- *CSS Reference – A free visual guide to CSS* https://cssreference.io/
- *CSS Reference* https://tympanus.net/codrops/css_reference/
- *DevDocs CSS Reference* https://devdocs.io/css/

További hasznos linkek (2)

- *CSS Zen Garden*
  http://www.csszengarden.com/
- *The State of CSS Survey*
  https://stateofcss.com/
  - *The State of CSS 2021*
    https://2021.stateofcss.com/