CSS: Stílus elkülönítése

Webprogramozás – 3. előadás

Sulyok Csaba
csaba.sulyok@gmail.com

1. rész

CSS: bevezető

Egymásba ágyazható stíluslapok (CSS)

- *Cascading Style Sheets*
- leírónyelv, melynek segítségével különböző stíluslapokat hozhatunk létre és ágyazhatunk be HTML oldalakba
- az egyes HTML elemek megjelenítési stílusát határozzák meg (méret, szín, betűtípus, háttérkép, stb.)
- a HTML standadizálásáért felelős *World Wide Web Consortium (W3C)* hozta létre
- verziók:
  - CSS1 (1996)
  - CSS2 (1998)
  - CSS3 (először 2014-ben jelent meg, modulonként fejlesztik, egyesek véglegesítve, másokon még dolgoznak
- előnyei:
  - egységes stílust biztosít a HTML dokumentumoknak (pl. ugyanazon honlapon, webalkalmazáson belül)
  - rövidebbé, átláthatóbbá teszi a HTML oldalakat
  - megjelenítési stílus és tartalom szétválasztása
  - rugalmasság (elég a stílusállományt módosítni)

CSS

- a CSS különálló nyelv a HTML-től
- általános formátum:

```
szelektor {
    kulcs: ertek;
}
```

- az újsor karakterek nem számítanak; a következő egyenértékű:

```
szelektor { kulcs: ertek; }
```

- több szelektor megadható, a beállítások mindre érvényesek:

```
szelektor1, szelektor2, szelektor3 {
    kulcs1: ertek1;
    kulcs2: ertek2;
}
```

- a komment formátuma is különböző: `/* comment */`

CSS kód elhelyezése

1. HTML elembeli inline `style` attribútum
   - csak egy elemre érvényes
   - a *szelektor* ebben az esetben maga az elem
   - nehezen olvashatóvá teszi a kódot, ezért ritkán használt

```
<div style="color:red">Én piros szöveg vagyok</div>
```

2. `style` elem a `head`-ben
   - jobb szétválasztás
   - hasznos ha csak egy HTML oldalra érvényes
   - hátrány ha több HTML állománnyal dolgozunk

```
<head>
    <!-- ... -->
    <style type="text/css">
        div { color: blue; }
    </style>
</head>
<body>
    <!-- ... -->
    <div>Én kék szöveg vagyok</div>
</body>
```

CSS kód elhelyezése

3. Külső CSS file betöltése a `head`-ben
   - legtisztább szétválasztás
   - újrahasznosítható
   - önmagában validálható: `https://jigsaw.w3.org/css-validator/`

`mystyle1.css`:

```
div {
    color: green;
}
```

HTML állomány:

```
<head>
    <!-- ... -->
    <link href="mystyle1.css" rel="stylesheet" type="text/css">
</head>
<body>
    <!-- ... -->
    <div>Én zöld szöveg vagyok</div>
</body>
```

CSS példa: külső állomány betöltése

```
...
<head>
    ...
    <link href="mystyle1.css"
          rel="stylesheet"
          type="text/css">

    <link href="./mystyle2.css"
          rel="stylesheet"
          type="text/css">

    <link href="http://kulsohost/style.css"
          rel="stylesheet"
          type="text/css">
    ...
</head>
...
```

- CSS könyvtárak, mint a **Bootstrap** egyszerű és elegáns CSS beállításokat és osztályokat nyújtanak.
- Külső könyvtárak elérhetőek egy *Content Delivery Network (CDN)*-ön: `https://cdnjs.org`
- Bootstrap tutorial: `https://www.quackit.com/bootstrap/tutorial/`

CSS

- egy HTML elem stílusa különbözőképpen (a leírt 3 szinten) adható meg, ezek végül egy egységes *"virtuális"* stílussá alakulnak (innen a *"cascading"* elnevezés)
- prioritási sorrend:
  - inline módon megadott stílus (legnagyobb prioritású)
  - belső (a head elemben megadott) illetve külső stíluslap
  - böngésző alapértelmezés szerint meghatározott stílusa

Konfliktushelyzetek kezelése

- Stílusdefiníció forrásai:
  - szerző
  - böngésző
  - felhasználó
- Stílusjegy fontossága
  - normál
  - fontos (pl. `p.right { align: right !important; }`)
- Prioritási sorrend konfliktus esetén:
  1. a felhasználó fontos beállításai (legnagyobb prioritás)
  2. a szerző fontos beállításai
  3. a szerző normál beállításai
  4. felhasználó normál beállításai
  5. böngészőből származó beállítások

Konfliktushelyzetek kezelése

- Specifikusság szerinti sorrend (továbbra is fennáló konfliktus esetén): (a prioritás csökkenő sorrendjében)
  1. `id` szelektor
  2. stílusosztály illetve pszeudo-szelektorok
  3. kontextussal megadott szelektorok (minél több elemtípus szerepel, annál specifikusabbnak számít)
  4. univerzális szelektor (`*` - minden elemre vonatkozik)
- amennyiben továbbra is konfliktus van:
  - a későbbi felülírja a korábban definiált stílusjegyet
- Hasznos hivatkozások:
  - CSS referencia (w3schools)
  - CSS specifikációk (W3C)

2. rész

CSS szelektorok

CSS szelektorok

Lehetséges szelektorok (prioritási sorrendben):

1. Adott **ID**-val ellátott elem
   - `#myid {...}`
   - hozzátartozó HTML: `<elemnev id="myid">...</elemnev>`
   - az ID-k legyenek **egyediek** a dokumentumon belül
2. Adott **osztály**ba tartozó elemek
   - `.className {...}`
   - hozzátartozó HTML: `<elemnev class="className">...</elemnev>`
   - több elem tartozhat ugyanazon osztályhoz
3. Valamely elemtípusra érvényes szelektor
   - `div {...}`
   - minden `<div>`-re érvényes (ha nem írja fölül valamelyiket fennebbi szelektor)
4. Szülő elemre érvényes szelektor
   - pl. a `body {...}` a teljes dokumentumtestre érvényes
   - ha egy kulcs nincs konkrétabban beállítva egy elemnek, örököl a szülőelemektől

CSS egyszerű szelektorok példa

```
<!DOCTYPE html>
<html>
    <head>
        <title>My first webpage with CSS in it</title>

        <!-- Location 2: style tag in HTML head -->
        <style type="text/css">
             /* Inside this style tag there is only CSS code. Notice even the comment syntax is different

            body {
                color: gray; /* set main text color to gray (unless overridden) */
            }
            div {
                color: blue; /* set color of all div elements to blue (unless overridden) */
            }
            .myclass {
                /* set color of all elements with class "myclass" to yellow" */
                /* notice color can also be set with RGB string */
                color: #cccc00;
            }
            #text2 {
                color: green; /* set color of element with ID of text2 to green */
            }
        </style>
    </head>
```

CSS egyszerű szelektorok példa

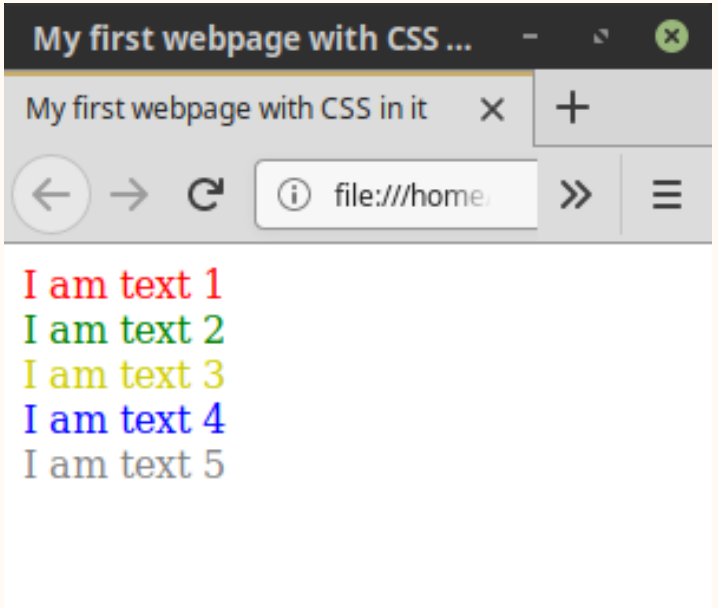
```
    <!-- ... -->
    <body>
        <!-- Priority 1 & Location 1:
             style given inline -->
        <div style="color:red">I am text 1</div>

        <!-- Priority 2: style given in header
             through ID -->
        <div id="text2">I am text 2</div>

        <!-- Priority 3: style given in header
             through class -->
        <div class="myclass">I am text 3</div>

        <!-- Priority 4: style given in header
             to all div tags -->
        <div>I am text 4</div>

        <!-- Priority 5: style given in header
             to body (no other styles applied) -->
        I am text 5
    </body>
</html>
```



CSS elhelyezkedés-alapú szelektorok

- `elem1 elem2` – az összes olyan `elem2`, mely leszármazottja egy `elem1` elemnek
  - pl. `div p {...};`
  - afferens HTML: `<div>...más elemek itt...<p>ide érvényes</p>...</div>`
- `elem1 > elem2` – az összes olyan `elem2`, mely direkt leszármazottja egy `elem1` elemnek
  - pl. `div > p {...};`
  - afferens HTML: `<div>...<p>ide érvényes</p>...</div>`
- `elem1 + elem2` – az összes olyan `elem2`, mely közvetlenül egy `elem1`-et követ
  - pl. `div + p {...};`
  - afferens HTML: `<div>...</div><p>ide érvényes</p>`
- `elem1 ~ elem2` – az összes olyan `elem2`, mely testvére egy `elem1`-nek (egy szinten van vele, nem feltétlenül közvetlenül követi)
  - pl. `div ~ p {...};`
  - afferens HTML: `<div>...</div>...más elemek itt...<p>ide érvényes</p>`

CSS attribútum szelektorok

- `[attributum] {...}` (pl. `[target]`) - kiválasztja az összes olyan elemet, mely **rendelkezik** `target` attribútummal
- `[attributum="ertek"] {...}` (pl. `[target="_blank"]`) - kiválasztja az összes olyan elemet, melynek `target` attribútumának értéke **egyenlő** `_blank`
- `[attributum^="ertek"] {...}` (pl. `[class^="top"]`) - kiválasztja az összes olyan element, melynek `class` attribútuma a `top` értékkel **kezdődik**
- `[attributum$="ertek"] {...}` (pl. `[class$="test"]`) - kiválasztja az összes olyan element, melynek `class` attribútuma a `test` értékkel **végződik**
- `[attributum*="ertek"] {...}` (pl. `[class*="te"]`) - kiválasztja az összes olyan elemet, melynek `class` attribútuma **tartalmazza** a `te` értéket
- több részlet

CSS pszeudo-osztályok

- egy elem pillanatnyi állapotát határozzák meg; interaktivitást biztosítanak.
- hozzájuk tartozó pszeudo-osztályokra írhatunk szelektorokat:

```
szelektor:pszeudoosztaly { kulcs: ertek; }
```

- negáló pszeudo-osztály: `:not(szelektor)`
- tipikusan hiperlinkeknél használt pszeudo-osztályok:
  - `a:link { color: red; }` - meg nem látogatott
  - `a:visited { color: blue; }` - már meglátogatott
  - `a:hover { color: green; }` - egérkurzor éppen fölötte
  - `a:active { color: lime; }` - aktív link
- elem elhelyezésére vonatkozó pszeudo-osztályok:
  - `:first-child { ... }` - első gyermeke a szülőelemnek
  - `:nth-child(n) { ... }` - *n*-edik gyermeke a szülőelemnek
- formelemek validálását szolgáló pszeudo-osztályok:
  - `:valid`, `:invalid`, `:required`, `:out-of-range`
- teljes pszeudoosztály lista

CSS komplex szelektorok

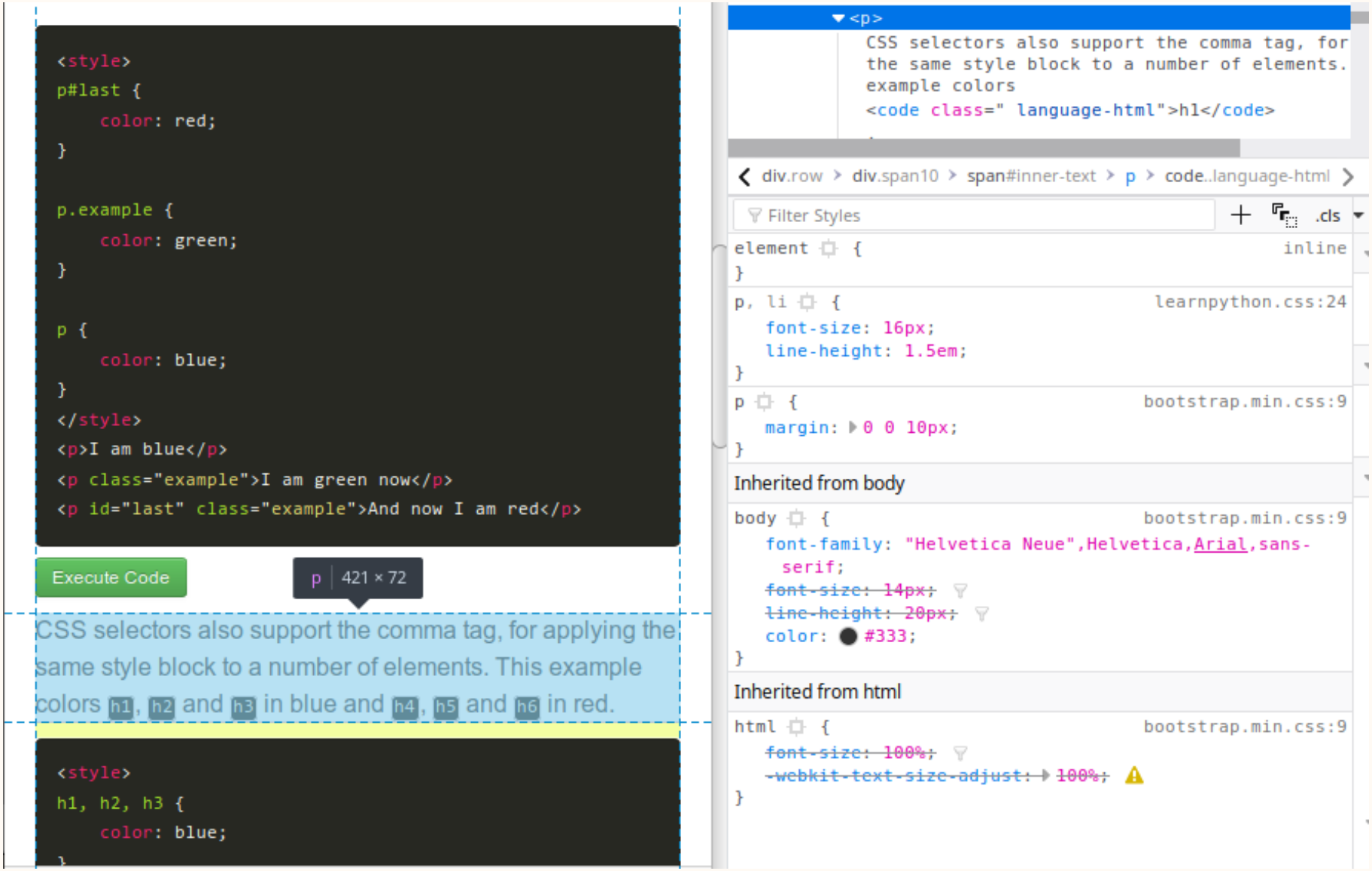
Több fent említett szelektort kombinálhatunk. Pár példa:

- `h1.className {...}`
  - az összes `h1` elem, mely bizonyos osztályba tartozik
- `a.highlight:hover { ... }`
  - az összes `a` elem, mely a `highlight` osztályba tartozik és egérrel rámegyünk
- `div:hover > p {...}`
  - olyan `p` elemek, melyek egy `div`-nek egyenes leszármazottjai és ezen `div`-re egérrel rámegyünk
- `input[type="text"]:invalid {...}`
  - az összeg szöveges úrlapmező, mely nem felel meg a validációs követelményeinek

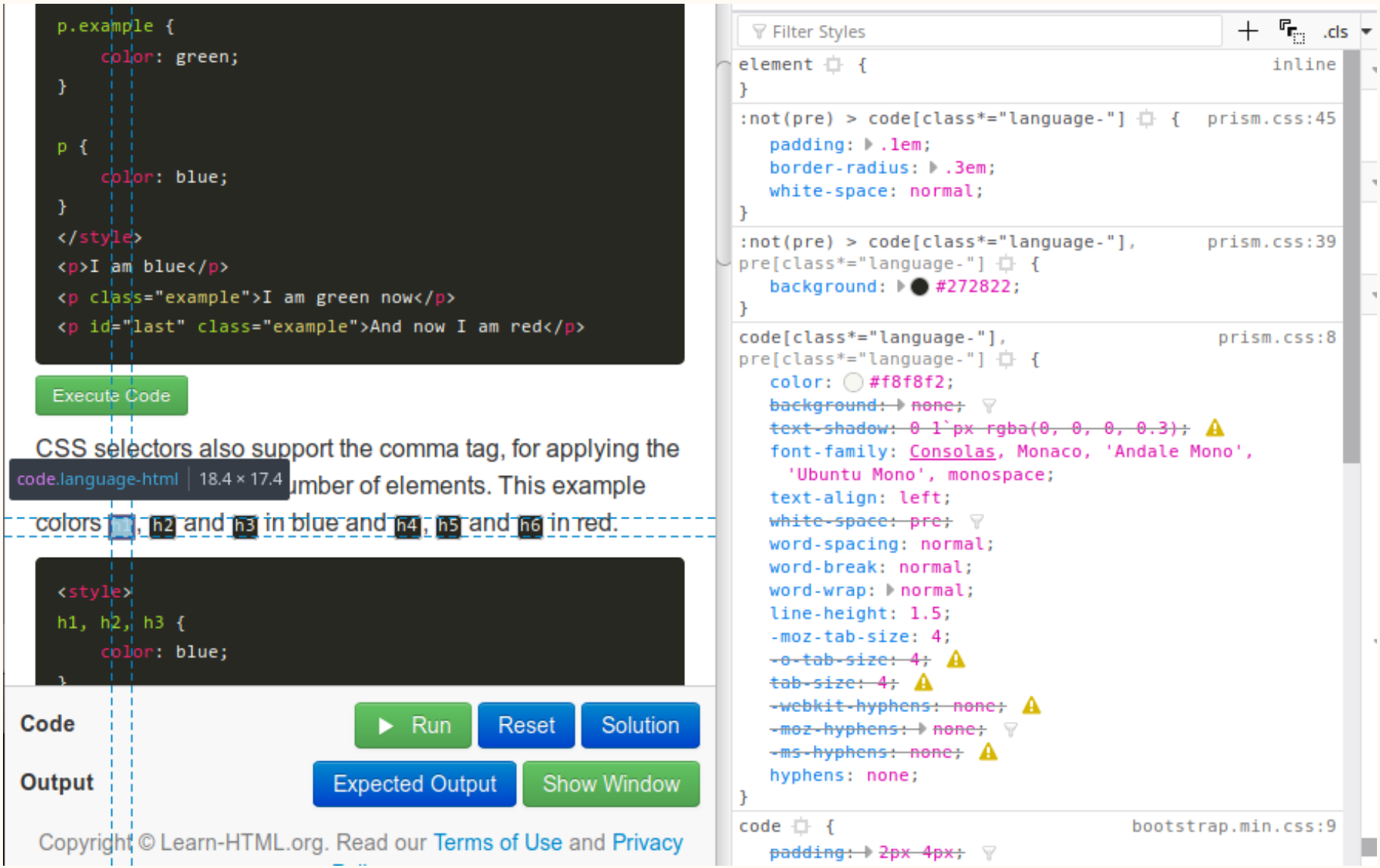
CSS inspekció

- Több szelektorban levő beállítás érvényes lehet egy bizonyos elemre. Ilyenkor az értelmező végigjárja a releváns beállításokat (**cascades**) és a *legspecifikusabbakat* alkalmazza.
- A böngészőben megnézhetjük, hogy egy-egy elemre milyen beállítások érvényesek, s melyik CSS beállításból származnak.

CSS inspekció



CSS inspekció



3. rész

CSS tulajdonságok

CSS tulajdonságok: bevezető

Gyakran használt CSS tulajdonságok:

- Szín/háttér tulajdonságok: `color`, `background-color`, `background-image`
- Szöveg tulajdonságok: `text-align`, `text-decoration`, `line-height`
- Betűtípus tulajdonságok: `font-family`, `font-style`, `font-weight`, `font-size`
- Elrendezés tulajdonságok: `margin`, `border`, `padding`, `display`

Teljes lista: `http://www.htmldog.com/references/css/properties/`

CSS tulajdonságok: háttér

- háttérszín: `body { background-color: #00ff00; }`
- háttérkép: `body { background-image: url("kep.gif"); }`
  - fix háttérkép: `body { background-image: url("kep.gif"); background-attachment: fixed; }`
  - szöveggel gördülő háttérkép (alapértelmezett)
  - kép ismétlése x, y, x-y tengely mentén, nincs ismétlés: `background-repeat: repeat-x, repeat-y, repeat, no-repeat`
  - háttérkép elhelyezése: `background-position`: `left/right/center/x%/xpos`, `top/bottom/center/y%/ypos`, esetleg `initial` vagy `inherit`
- rövidítés lehetősége
  - amennyiben több tulajdonságnév ugyanazzal az előtaggal kezdődik, rövidítve is megadhatjuk ezeket
  - ismerni kell ezek megadásának sorrendjét
  - pl. `background` esetében a sorrend:
    - `background-color`
    - `background-image`
    - `background-repeat`
    - `background-attachment`
    - `background-position`

```
body { background: #00ff00 url("smiley.gif") no-repeat fixed center;
```

CSS tulajdonságok: Színek

A színek megadhatóak:

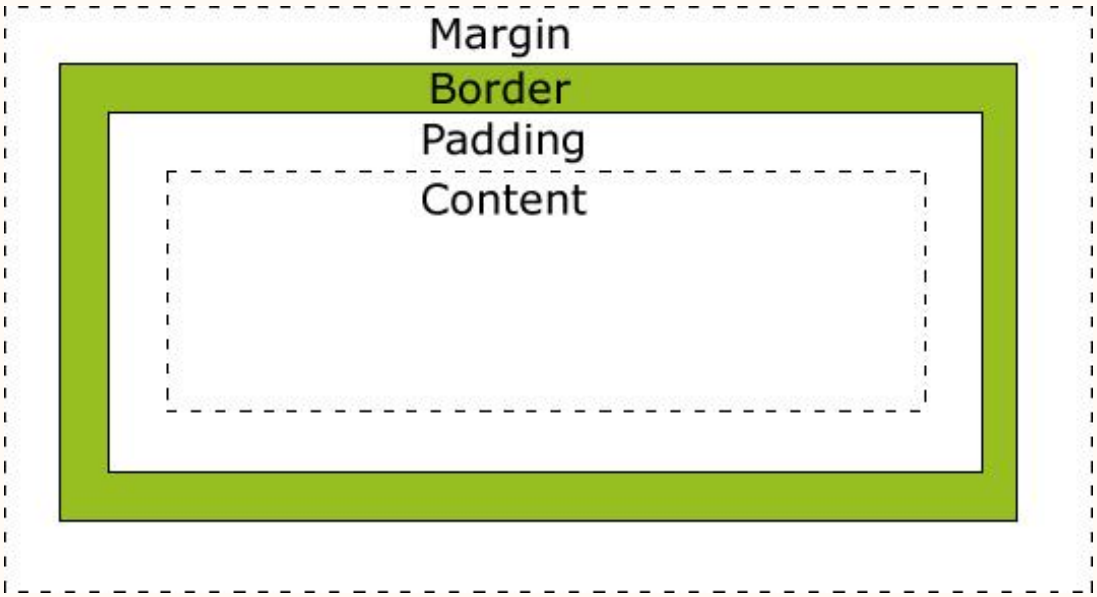
- numerikusan RGB hexakódokkal (`color: #AABB00`)
- szimbolikusan (`color: red`)
- RGB komponensek segítségével (`color: rgb(255,0,0)`)
- RGBA színek (`color: rgba(0, 255, 0, 0.3)`)
- HSL (hue-saturation-lightness) színek (`color: hsl(120, 100%, 25%)`)
- HSLA színek (`color: hsla(120, 60%, 70%, 0.4)`)

Játék a színekkel:

- Színek és árnyalatok választása - a w3schools oldalán
- Összeillő színárnyalatok generálása

CSS dobozmodell (box model)

Szél (`margin`), szegély (`border`), behúzás (`padding`) (W3C specifikációja)



megj.:

- egy elem `width` illetve `height` tulajdonságai a tartalomra (content) vonatkoznak

Használható mértékegységek

- Abszolút hossz
  - `cm` (centiméter), `mm` (milliméter)
  - `px` (pixel) - *nem felel meg az eszköz pixeleinek*
  - `in` (inch; 1 in = 96 px = 2.54 cm))
  - `pt` (pont; 1 pont = 1/72 in ∼ 0.0353 cm)
  - `pc` (pica; 1 pica = 12 pont)
- Relatív hossz (inkább ajánlott a használata)
  - `em` - az aktuális elem betűméretéhez viszonyított érték (pl. `2em` az aktuális betűtípus kétszerese)
  - `ch` - a “0” karakter szélességéhez viszonyítitt
  - `rem` - a gyökér-elem betűméretéhez viszonyított
  - `vw` - a nézetablak (viewport) szélességének 1%-ához viszonyított
  - `vh` - a nézetablak (viewport) magasságának 1%-ához viszonyított
  - `%` (százalékban megadott érték)

Szegély – `border`

- tulajdonságok
  - szegély stílusa – `border-style` (kötelező megadni): pl. dotted, dashed, solid, double, groove (3D), ridge (3D), inset (3D), outset (3D), none, hidden
  - szegély vastagsága – `border-width`
  - szegély színe – `border-color`
  - kerekített szegély – `border-radius`, pl. `border-radius: 8px`
- külön állíthatóak a felső, jobboldali, alsó, baloldali rész tulajdonságai (pl. `border-top-style`). *Sorrend:* top, right, bottom, left
- tulajdonságok rövidítése: `*-width`, `*-style`, `*-color`, pl.

```
p {
    border-left: 6px solid red;
    background-color: lightgrey;
}
```

Szövegformázás

- szöveg színe - `color`
- szövegigazítás - `text-align`: right, left, center, justify
- szövegdíszítés - `text-decoration`: overline, underline, none, line-through
- módosítás - `text-transform`: lowercase, uppercase, capitalize
- behúzás - `text-indent` (pl. `5px`)
- betűtávolság - `letter-spacing` (pl. `2px`, `-2px`)
- sormagasság - `line-height` (0.8, 1.8)
- szöveg irányítása - `direction`: ltr, rtl
- szavak közti távolság - `word-spacing` (pl. 10px, -5px)
- árnyék (vízszintes, függőleges méret, szín) - `text-shadow` (pl. 3px 2px red)

Betűtípus

- betűtípus család (`font-family`)
  `css p { font-family: Verdana, Arial, 'Times New Roman', serif }`
- általános típusok: serif, sans-serif, cursive, fantasy, vagy monospace
- betű stílus (`font-style`): *italic*, *normal*
- betűvastagság (`font-weight`): *normal*, *bold*, *bolder*, *lighter*, *100*, ..., *900*
- betűméret (`font-size`): abszolút érték (pl. x-small, small, medium, large), relatív (pl. larger, smaller), hossz, % (százalék - a szülő elem betűméretéhez viszonyított)

Lásd még:

- gyakran használt betűtípusok
- Google fonts

Megjelenítés

- megjelenítés (`display`) módosítása:
  - `display: inline;`
  - `display: block;`
- láthatóság (visibility):
  - `display: none;`
  - `visibility: hidden;`

Pozíció megadása:

- `position`:
  - statikus (`static`) - alapértelmezett
  - fix (`fixed`) - elhelyezés a böngészőablakhoz képest (a görgetés sem befolyásolja az elhelyezést)
  - relatív (`relative`) - a normál elhelyezéshez viszonyított relatív elhelyezés
  - abszolút (`absolute`) - az első nem statikus pozíciójú szülőelemhez viszonyított elhelyezés
- pozíció megadása: `top`, `right`, `bottom`, `left` tulajdonságok segítségével
- `z-index`:
  - egymást átfedő elemek esetén megadható a megjelnítés sorrendje
  - nagyobb index-értékű elemek a kisebb index-értékűek fölött helyezkednek el

float/clear tulajdonságok

- a **float** tulajdonság segítségével
  - jobb vagy baloldalra tolhatunk egy elemet
  - az utána következő elemek körülötte fognak elhelyezkedni
- a **clear** tulajdonság
  - megakadályozza, hogy az elem bal vagy jobb (esetleg mindkét) oldalán balra/jobbra “folyatott” elem helyezkedjen el
- tipikus használat
  - szöveg, illetve kép(ek) egymás mellé rendezése
  - weboldal tartalmának részekre bontása, és ezek elhelyezése táblázat használata nélkül (Nem ajánlott. használjunk inkább **flexbox**ot)

4. rész

CSS3

CSS3

- A jelenleg legújabb CSS specifikáció verzió
- Első hivatalos újítás 1998 óta
- Moduláris fejlesztés
- Főbb újdonságok:
  - tartalom-specifikus szelektorok (attribútumok segítségével);
  - transzformációk - pl. elemek 90deg-os elfordítása;
  - dinamikus átmenetek változó elemeknek;
  - responsive web design *media queries* - tartalom és elosztás módosítása képernyőméret és típus szerint (mobilbarát)

CSS3 Media query

Media query

- lehetővé teszi a tartalom megjelenítésének testreszabását, attól függően, hogy milyen kliens-készülékről van szó, anélkül, hogy a tartalmat meg kelljen változtatni
- általános alak:
  `@media not|only` *médiatípus* `and *kifejezések* {`*stílus*`}`

```
@media print {
    body { font-size: 10pt }
}
@media screen {
    body { font-size: 13px }
}
@media screen, print {
    body { line-height: 1.2 }
}
```

CSS3 Media query

Média típusok

- **all** - minden eszköz
- **print** - nyomtató
- **screen** - képernyő
- **speech** - képernyőolvasó

Kiértékelés: Amennyiben a *médiatípus* megegyezik a kliens eszköz típusával, és minden logikai kifejezés, ami szerepel a lekérdezésben, igaz, akkor a lekérdezés törzsében levő CSS szabályok alkalmazva lesznek.

CSS3 Media query

Néhány példa különböző hordozható eszköz lekezelésére:

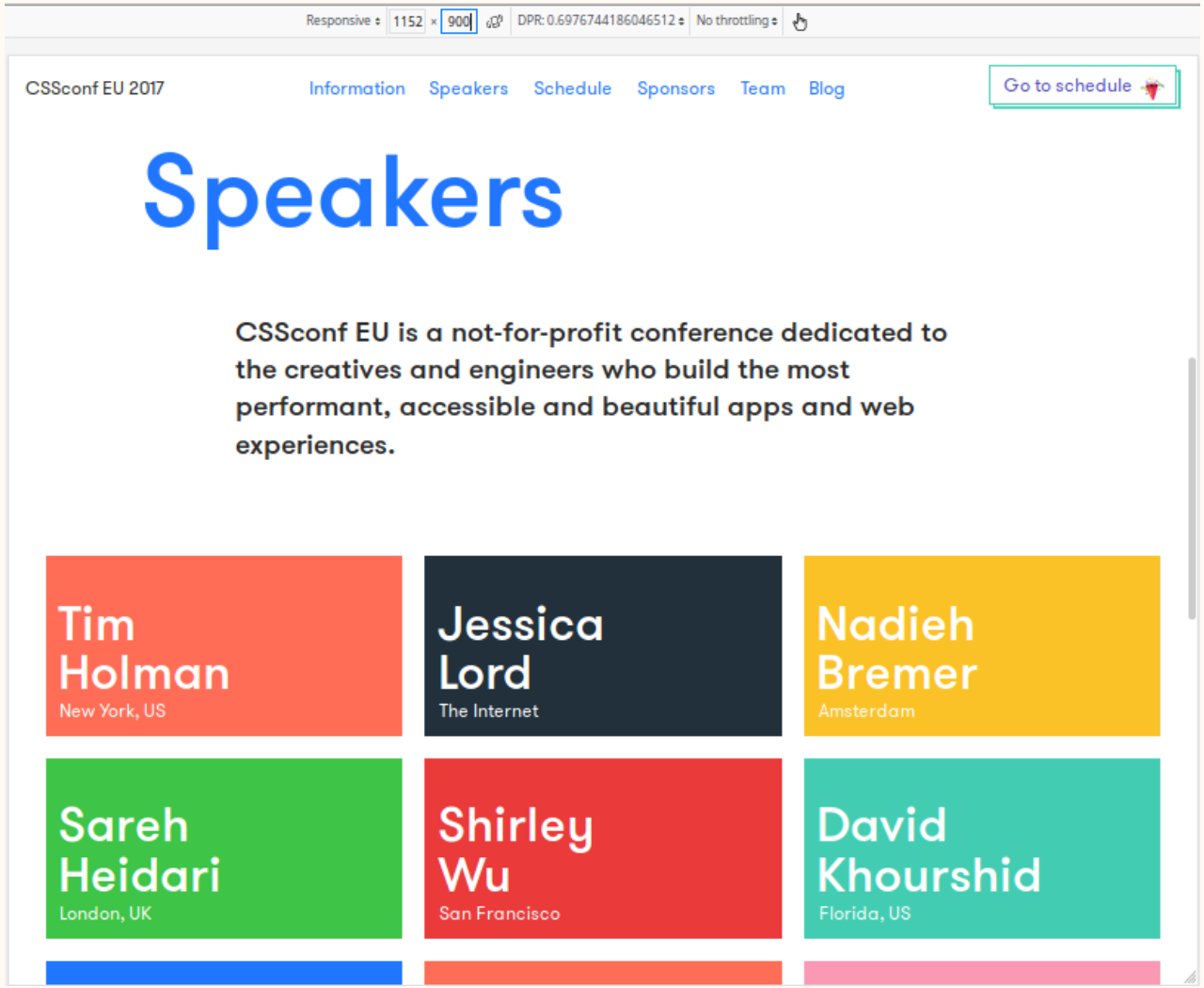
```
/* iPhone 5 ----------- */
@media only screen
  and (min-device-width: 320px)
  and (max-device-width: 568px)
  and (-webkit-min-device-pixel-ratio: 2) {
    /* Stilusjegyek */
}
/* iPad 3 vagy 4 (landscape) ----------- */
@media only screen
  and (min-device-width: 768px)
  and (max-device-width: 1024px)
  and (orientation: landscape)
  and (-webkit-min-device-pixel-ratio: 2) {
    /* Stilusjegyek */
}
```

CSS3 Media query

Példa: a képernyőszélesség függvényében változtatjuk egy elem tulajdonságait:

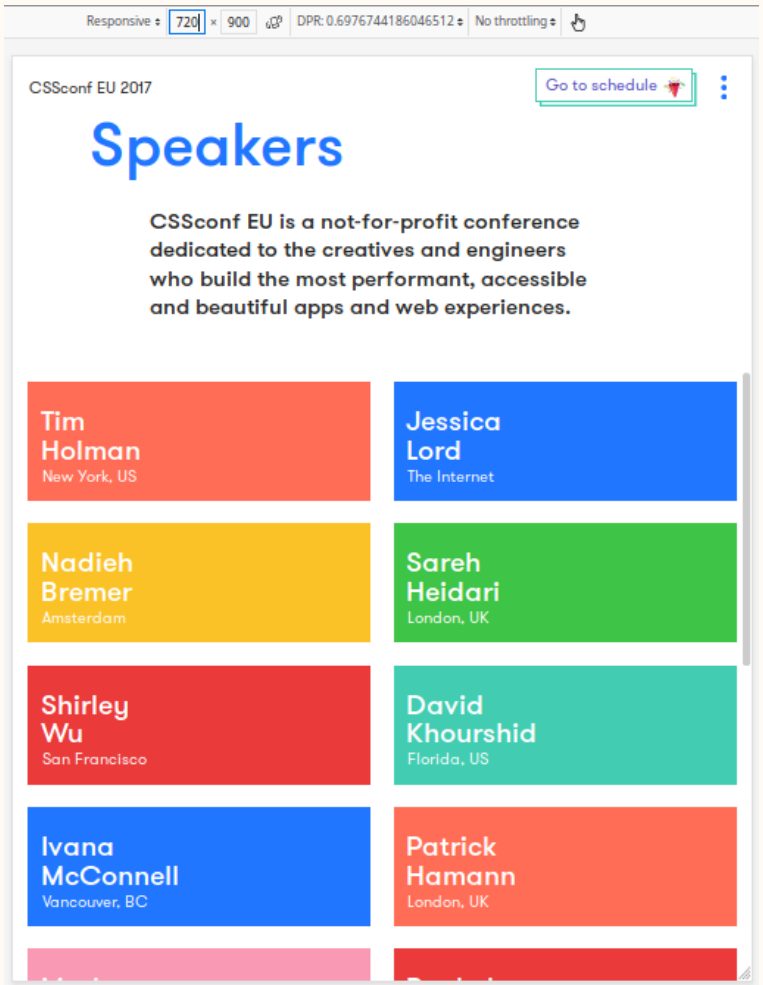
```
@media screen and (min-width:600px) {
    nav {
        float: left;
        width: 25%;
    }
    section {
        margin-left: 25%;
    }
}
@media screen and (max-width:599px) {
    nav li {
        display: inline;
    }
}
```

CSS3 Media query példa

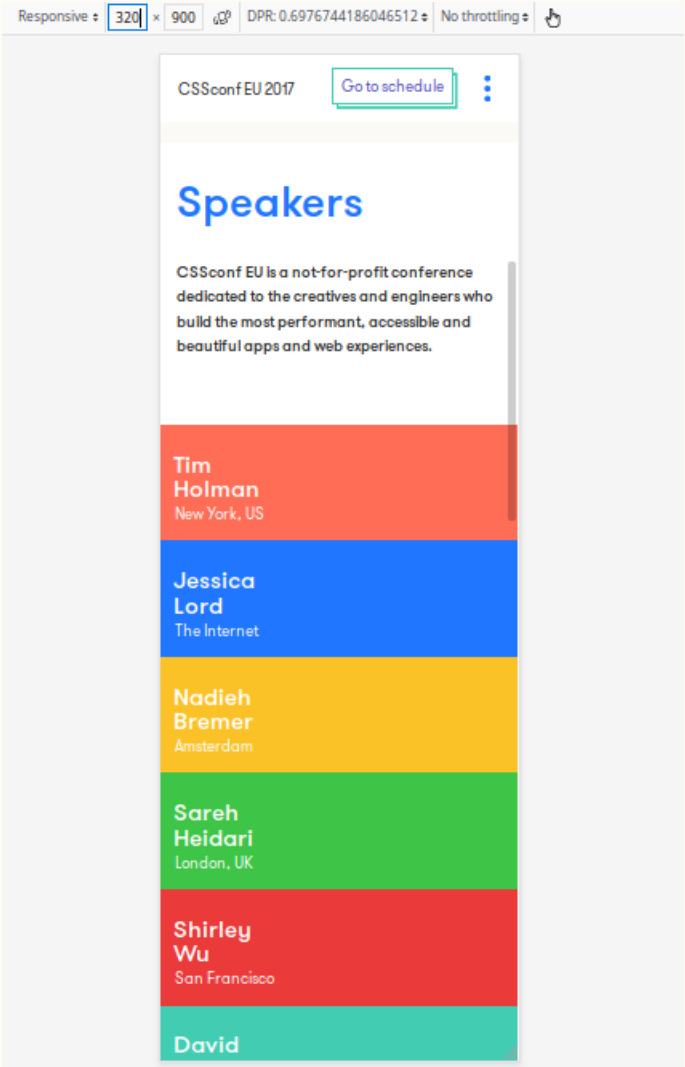


képernyőszélesség: 1152px

CSS3 Media queries



képernyőszélesség: 720px



képernyőszélesség: 320px

CSS3 átmenetek (tranziciók)

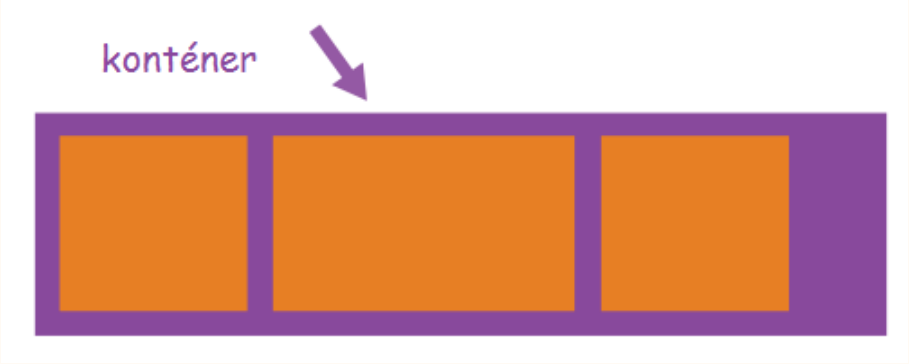
- CSS átmenetekkel meghatározhatjuk annak a módját, ahogyan egy adott elem valamilyen stílusjegye megváltozik.
- Két dologot kell definiálnunk a *transition* tulajdonság segítségével:
  - a **tulajdonság** amelyre szeretnénk az átmenetet alkalmazni
  - az átmenet **időtartalma**
- Az átmenet akkor veszi kezdetét, amikor a megjelölt tulajdonság *megváltozik*. Amennyiben időtartamot nem adunk meg, ez alapértelmezetten 0, tehát az átmenet nem fog látszani.

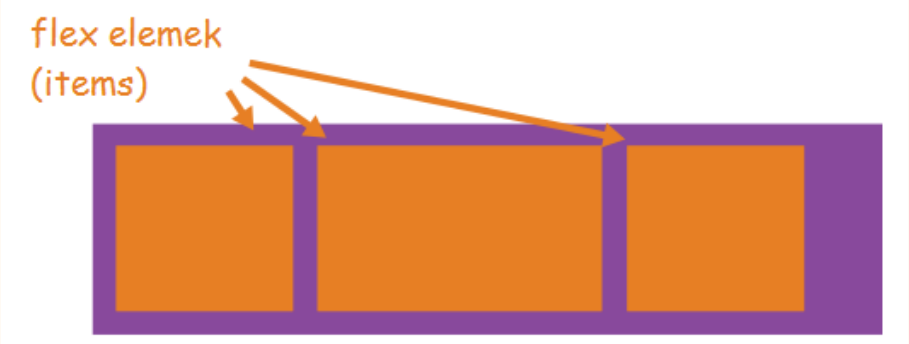
```
div#transitiondiv {
    background-color: green; /* a tulajdonsag eredeti erteke */
    transition: background-color 3s; /* tulajdonsag es idotartam megadasa */
}
div#transitiondiv:hover {
    background-color: blue; /* kivaltja az adott tulajdonsag valtozasat */
}
```

CSS3 flexible box layout model

- Specifikáció: `https://drafts.csswg.org/css-flexbox/`
- Beosztás és elrendezés optimalizálása olyan esetekben, amikor előre ismeretlen egy-egy elem mérete, s nem elegendő a block (“szülőelem maximális szélessége”) és inline (“gyerekelemek minimális szélessége”) fogalmak finomsága
- Elemek elrendezhetőek vízszintesen és függőlegesen is

Terminológia:





CSS3 flexible box layout model

konténer-tulajdonságok:

- flex-konténer **display**: `flex|inline-flex`
- **flex-direction**: `row | row-reverse | column | column-reverse` – főtengely meghatározása, melynek mentén az egyes flex-elemek el lesznek helyezve
- **flex-wrap**: `nowrap | wrap | wrap-reverse` – eredetileg egy sorban próbálnak elférni az elemek, ezzel a tulajdonsággal módosíthatjuk ezt
- **justify-content**: `flex-start | flex-end | center | space-between | space-around` – elemek igazítása a főtengely mentén
- **align-items**: `flex-start | flex-end | center | baseline | stretch` – elemek igazítása a kereszttengely mentén
- **align-content**: `flex-start | flex-end | center | space-between | space-around | stretch` – amennyiben több sorban helyezkednek el a flex-elemek, meghatározza ezek igazítását

CSS3 flexible box layout model

flex-gyerekelem tulajdonságok:

- **flex-grow**: `<pozitív szám>` (alapértelmezetten 0) - megadhatjuk, hogy az egyes elemek egymáshoz viszonyítva milyen arányban növekedhetnek, ha van hely
- **flex-shrink**: `<pozitív szám>` (alapértelmezetten 0) - megadhatjuk vele a lehetőséget egy flex-elemnek, hogy zsugorodjon
- **flex-basis**: `<hossz> | auto` - megadja egy elem méretének alapértelmezett értékét, mielőtt a megmaradt hely elosztása megtörténne
- **align-self**: `auto | flex-start | flex-end | center | baseline | stretch` - egyes elemek igazítását megadhatjuk önállóan (az `align-items`-ben megadottól eltérő lehet)

CSS3 flexbox példa

```html
<body>
<header>
  <h1>Using flexboxes</h1>
</header>
<div id="container">
  <nav>Navigation</nav>
  <main>
    <section>...</section>
  </main>
  <aside>Aside</aside>
</div>
<footer>...</footer>
</body>
```

```css
body {
  display: flex;
  flex-direction: column;
}
#container {
  display: flex;
  flex-direction: row;
}
nav, aside {
  flex: 1 1 20%;
}
main {
  flex: 3 3 60%;
}
```

```css
@media (max-width: 600px) {
  #container {
    flex-direction: column;
  }
  main { order: 3; }
  nav { order: 1; }
  aside { order: 2; }
  main, nav, aside {
    flex: 0 0 auto;
  }
}
```