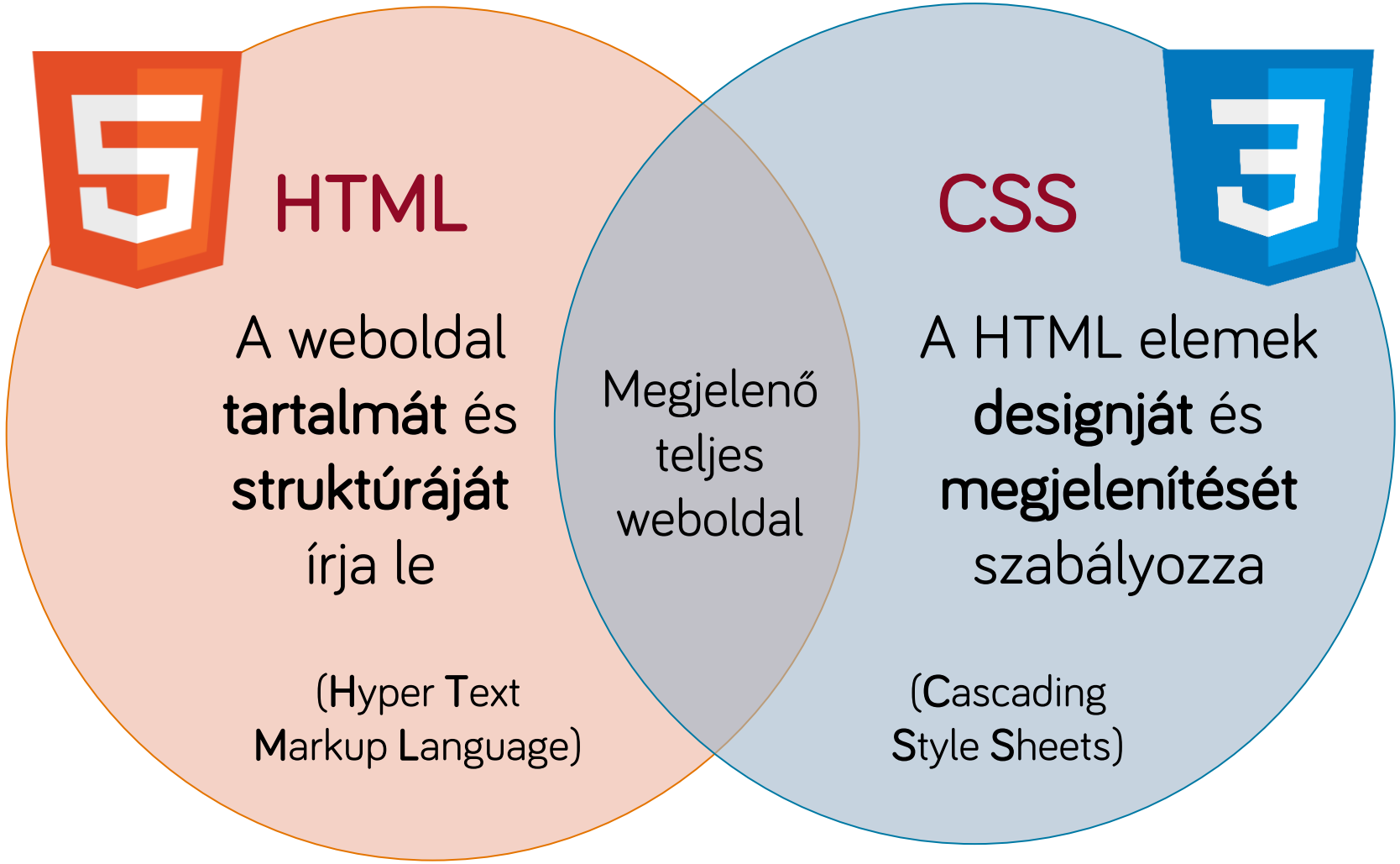
CSS bevezetés

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

A HTML és a CSS kapcsolata



A CSS egyszerű nyelvnek tűnik...

- Kulcs-érték párokat állítunk be fix értékkészletből olyan magától értetődő tulajdonságokra mint szín vagy háttér.
- A szabályokat deklaratívan adhatjuk meg a dokumentum adott részeire vonatkozólag.
- Amíg csak szöveget formázunk, minden rendben.
- Amint layoutot szeretnénk definiálni, vagy amint pixelpontos dizájn kell, bonyolódik a helyzet.

CSS alapok

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Inline stílusok

- A HTML <head> tagjében hozhatunk létre stílusokat melyekben
  - Kulcs – érték párokat állítunk be az oldal egyes elemeire
- Ezek a stílusok adják meg az oldal designját.
- Az inline stílusokat azért nem szeretjük, mert a HTML a tartalomért felelős nem a designért.
  - Szervezzük ki a stílusokat külön fájlokba

```
<head>
    <style>
        h1 {
            color: #A4001C;
            font-size: 26px;
        }
    </style>
</head>
```

index.html

```
<head>
    <link rel="stylesheet" href="style.css">
</head>
```

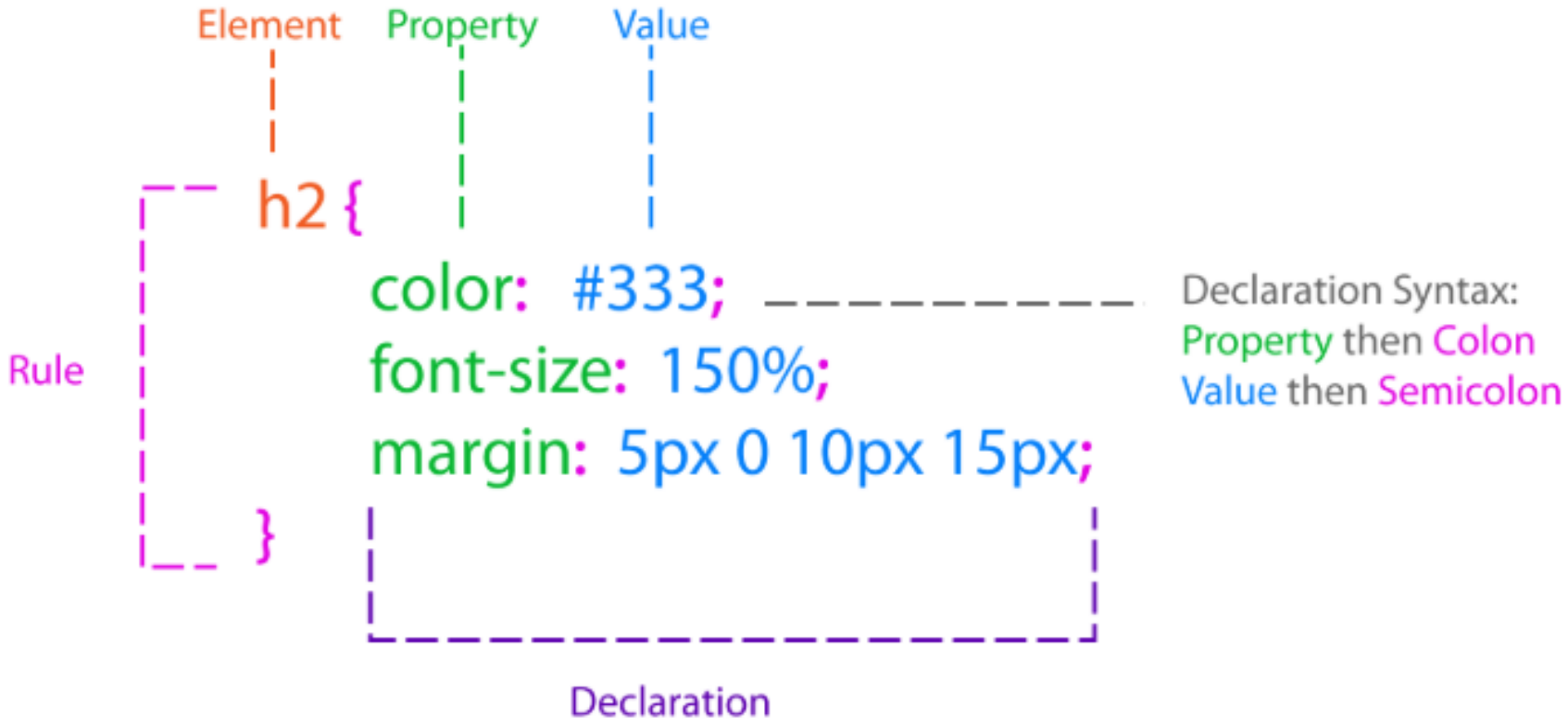
style.css

```
h1 {
    color: #A4001C;
    font-size: 26px;
}
```

Alapok

- A HTML oldal egy vagy több CSS fájlt is hivatkozhat.
- A szabályok **selectorral** kezdődnek
  - megadja, milyen elemekre kell őket alkalmazni.
- A selectoron belül kulcs-érték párokat tartalmaznak a tulajdonságok beállítására.
- Egyes selectorok magasabb prioritásúak másoknál, és felülírhatják egymás szabályait.

CSS szabályok felépítése



CSS példa

```
h2 {
      color: #A4001C;
      font-size: 26px;
}
/* Minden link legyen bordó */
a {
      color: #A4001C;
}
/* Kivéve ami a nav-ban van, mert az fekete */
nav a { /* A specifikusabb selectorok erősebbek */
      color: black;
}
```

Egyszerű CSS selectorok

CSS selector	HTML Tag
Tetszőleges HTML tagre	

`a {`
`  color: red;`
`}`
	`<a>`
	`  …`
	`</a>`

Saját osztályokra, amire a class attribútummal hivatkozunk	

`.osztaly{`
`  color: red;`
`}`
	`<div class="osztaly">`
	`  …`
	`</div>`

Tetszőleges azonosítóra, amire az id attribútummal hivatkozunk	

`#egyedi_azonosito{`
`  color: red;`
`}`
	`<div id="egyedi_azonosito">`
	`  …`
	`</div>`

Forrás: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Attribútum CSS selectorok

CSS selector	HTML Tag
Ha létezik az attribútum	

`a[href] {`
`  color: red;`
`}`
	`<a href="…">`
	`  …`
	`</a>`

Ha az attribútum értéke…	

`a[href="https://www.aut.bme.hu"]{`
`  color: red;`
`}`
	`<a href="https://www.aut.bme.hu">`
	`  …`
	`</a>`

- Ha a href tartalmazza az, hogy aut (szótöredéket): `a[href*="aut"]`
- Ha a href azzal végződik, hogy bme.hu: `a[href$=".bme.hu"]`
- Ha a class tartalmazza, hogy a logo (egész szót): `a[class~="logo"]`

Forrás: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Selectorok összefűzése (szóköz nélkül)

- A tag, osztály, Id és attribútum selectorokat szóköz nélkül egymásután írva olyan selectort kapunk, ami csak azokat az elemeket választja ki, amire mindegyik rész igaz.
  - Pl.: Azok az <img> tagek, amin van src attribútum és rajta van a selected class

```
img[src].selected { border: 1px solid red; }
```


Forrás: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Hierarchikus selectorok

CSS selector	HTML Tag
Tag alatt közvetlenül osztály (**Közvetlen leszármazottak**)	

`nav > .main {`
`  color: red;`
`}`
	`<nav>`
	`  <a class="main">…</a>`
	`</nav>`

Tagen belüli osztály (összes **leszármazottak**)	

`nav .main{  // Figyeljünk a szóközre`
`  color: red;`
`}`
	`<nav>`
	`  <ul><li class="main">…</li></ul>`
	`</nav>`

Utána következő **Testvér** elemek kiválasztása	

`img ~ p {`
`  color: red;`
`}`
	`<p>Ezt nem jelöli ki</p>`
	`<img src="…">`
	`<p>Ezt jelöli ki</p>`
	`<p>És ezt is</p>`

Közvetlenül utána következő **Testvér** elem kiválasztása	

`img + p {`
`  color: red;`
`}`
	`<p>Ezt nem jelöli ki</p>`
	`<img src="…">`
	`<p>Ezt jelöli ki</p>`
	`<p>Ezt már nem</p>`

Forrás: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Univerzális selector

- Az univerzális selector segítségével az oldalon lévő összes elemet kijelölhetjük
  - Pl: minden elem kerete legyen 1 pixeles szaggatott piros vonal.

```
* { border: 1px dashed red;}
```

- Az univerzális selectort
  - nagyon ritkán használjuk.
  - nagyon gyenge selector, szinte minden szabály felülírja
  - Ha debugoláshoz használjuk, akkor a !important-ot érdemes rátenni.

Forrás: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Pszeudo osztályok

Az egyes elemek állapota alapján tudunk megjelenést módosítani.

- Meglátogatott link:

```
:visited { color: red;}
```

- Letiltott inputok:

```
:disabled { color: gray;}
```

- Csak olvasható inputok:

```
:readonly { color: gray;}
```

- Első gyerek:

```
:first-child { color: orange;}
```

- Ha fölötte van az egér:

```
:hover { color: green;}
```


Forrás: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Pszeudo elemek

- A kiválasztott elem egy részét lehet vele testreszabni.
  - Új HTML elem létrehozása első gyerekként

```
section::before { content: ''; }
```

  - Új HTML elem létrehozása utolsó gyerekként

```
section::after { content: ''; }
```

  - Kiválasztott elemek (pl.: szövegrész) formázása

```
::selection { background-color: yellow; }
```

  - Első betűre egyedi megjelenés

```
::first-letter{ font-size: 40px; }
```


Forrás: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Blokk vs inline elemek

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Az elemek két* fő típusa

- Inline
  - Amik „egymás mellé kerülnek”
  - Pl.: `span, a, img,` stb.
- Blokkszerű
  - Amik „új sort kezdenek”
  - Pl.: `div, form` stb.

Blokk elemek tulajdonságai

- Mindig új soron kezdődik, és a szülője teljes szélességét kitölti.
  - Az egymás után következő blokk elemek – ha a stílus felül nem írja – egymás alá rendeződnek.
  - Ez az ún. "block flow", ennek iránya mindig fentről lefele.
  - A blokkok egymásba ágyazhatók, szülő-gyerek viszonyt létrehozva, de ezen belül is a block flow érvényesül.
    - a gyerekek egymás alá rendeződnek, fentről lefelé, mindig új sort kezdve

Inline elemek tulajdonságai

- Az inline elemek egy soron belül követik egymást
- Leginkább szövegfolyamba illő elemekről van szó.
  - pl. linkelt szöveg, hangsúlyos szöveg, vagy mondjuk egy emoji…
  - a szövegbe "belesimulnak", nem kezdenek új sort
  - Következésképp a folyam iránya vízszintes, és ~~balról jobbra~~ követi a dokumentum szövegirányát. (i18n)

Az egyes típusok által felvehető tulajdonságok

- Az imént említett viselkedés csak egy alapértelmezés, szinte minden felülírható.
- Hogy egy elem block vagy inline jellegű-e, nem csak a dokumentumfolyambeli viselkedésre van hatással.
  - > Befolyásolja a rá érvényesíthető doboz modellt
  - > Befolyásolja az általa felvehető CSS tulajdonságok halmazát

Inline elemek

- Nem reagálnak a szélesség, a magasság és a függőleges padding / margó beállítására.
  - Elvileg annak érdekében, hogy a szövegfolyam „ne törjön meg” nagyon, de pl. képekkel ezt el lehet rontani
- Reagálnak a
  - vízszintes rendezésre (`text-align`)
  - sormagasság beállítására (`line-height`)
  - és az azon belüli rendezésre (`vertical-align`)
- Blokk elemekre a `text-align, line-height, vertical-align` tulajdonságok nem hatnak, de az inline gyerekelemeik megöröklik, így közvetett hatása van.

Inline elemek „blokkosítása”

- Bármely inline elem blokk elemmé alakítható a `display` tulajdonság `block`-ra állításával
- Ezen felül egyes CSS propertyk "kiszakítják" az inline elemeket a dokumentumfolyamból, *effektíve* blokkszerűvé alakítva őket
  - pl. `float`, `position`…
- Így a `width`, `height` stb. beállíthatóvá válik rajtuk.

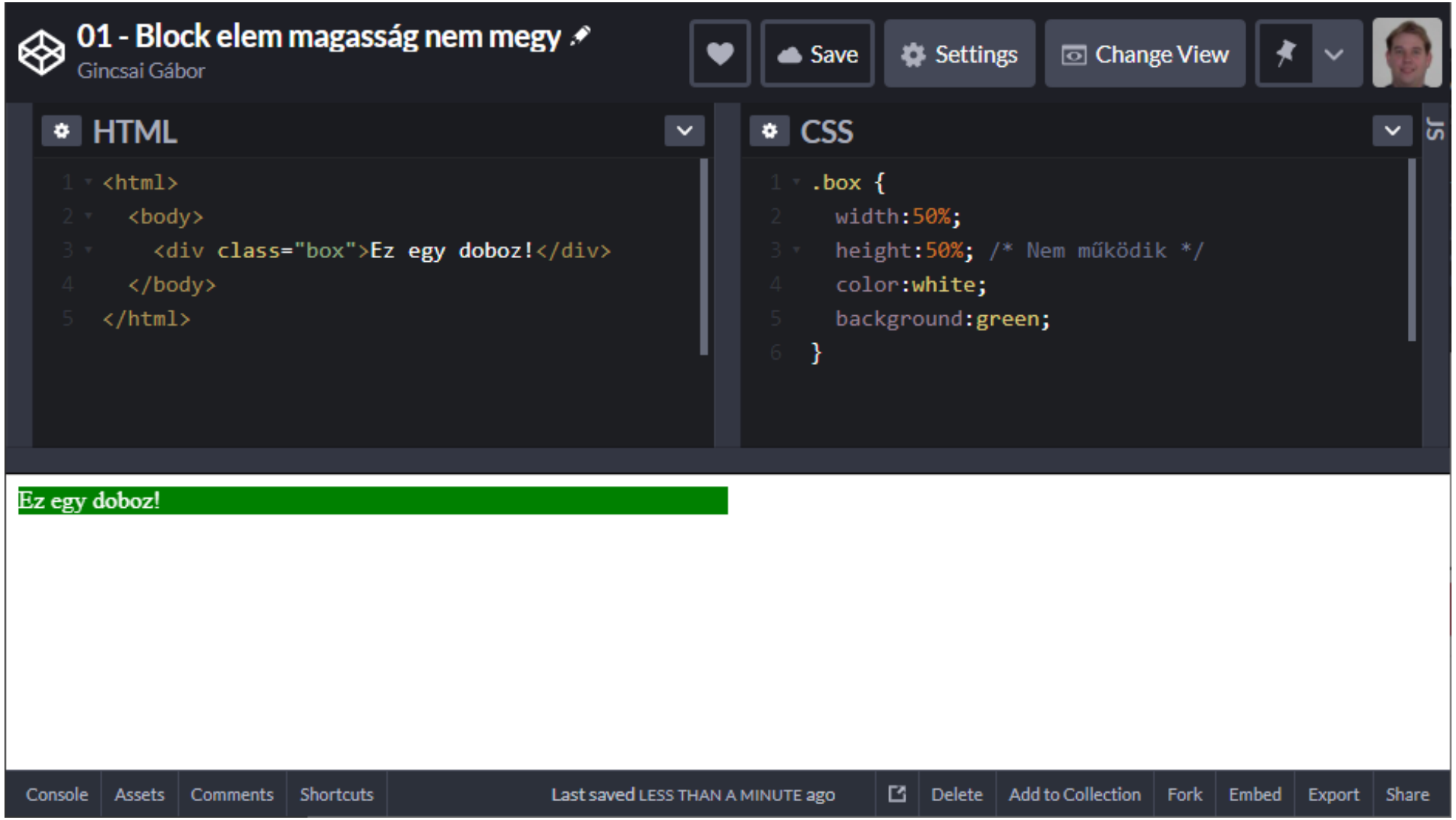
Blokk elemek – width

- Alapértelmezése `auto`, ennek hatására a blokk kitölti a szülője szélességét.
- A legtöbb CSS mértékegységet megeszi, így pl. `px`, `%`, `em`, `rem`, `vw`, `vh`…
  - A `%` a szülő szélességének %-ában értelmezendő
  - Az `em` a szülő betűméretéhez, a `rem` a gyökér (`html`) betűméretéhez képest relatív
  - A `vw` és a `vh` a viewport aktuális méretéhez képest relatív

Blokk elemek - height

- A `width`-hez hasonlóan a height-nak is `auto` az alapértelmezése, és mindenféle CSS hosszúság beállítható rajta
  - De: az `auto` jelentése: a ***tartalmazott elemek*** megjelenítéséhez szükséges magasság
  - Tehát nincs helykitöltés, mint a szélesség esetén
- A % a szülő magasságának %-ában értendő
  - de ez **nem mindig adja a várt eredményt**

Blokk elem magassága miért nem 50%?



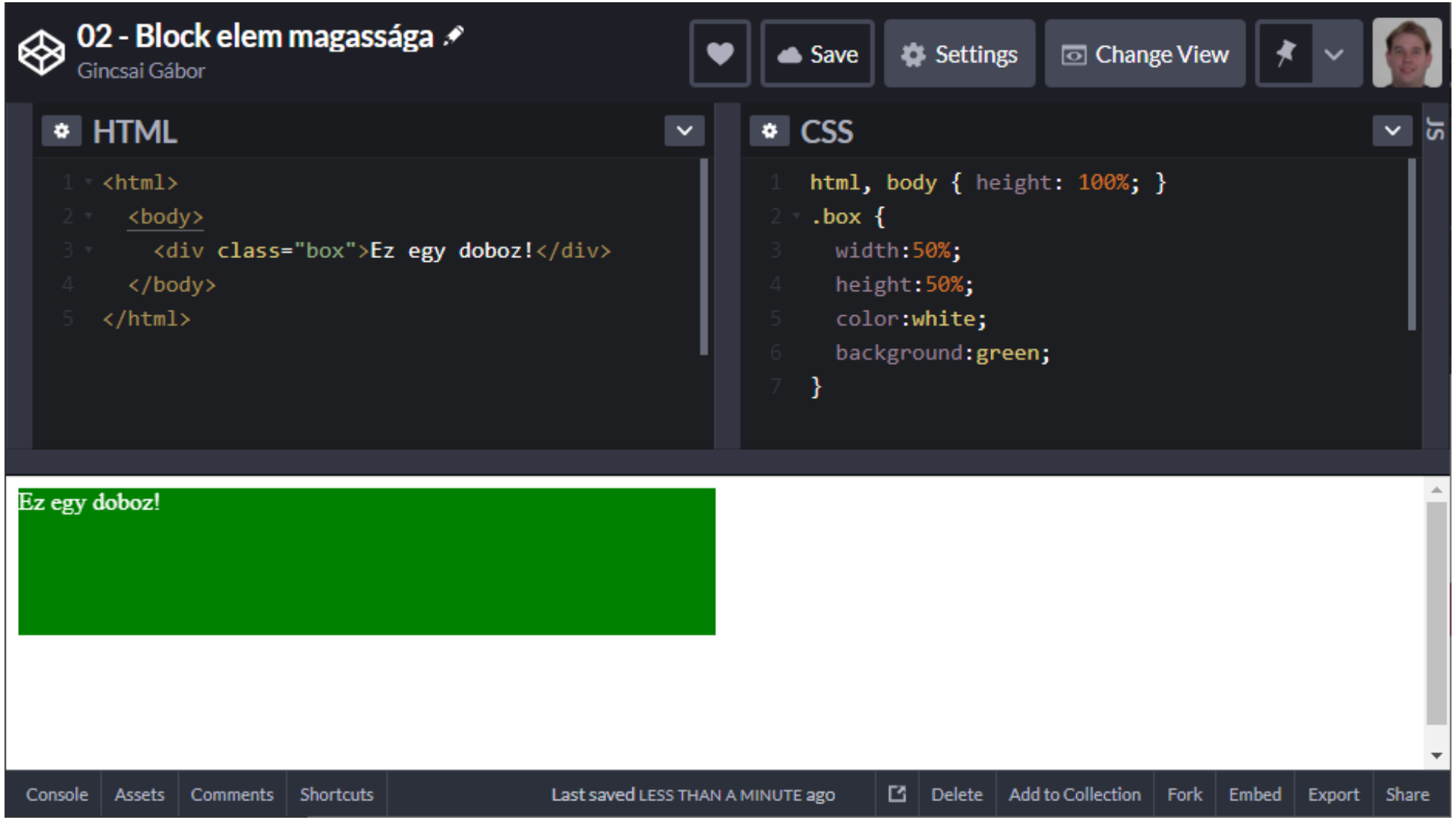
https://codepen.io/gincsai/pen/rGqbJQ

Miért nem működik?

- Mit adtunk meg?
  - Beállítottuk a dobozon az 50% magasságot.
  - Szülőnek nincs megadva magasság → auto
    - a szülő magassága éppen elegendő az elem tartalmának megjelenítéséhez.
  - E miatt a szülő kisebb lesz.
- A fán fölfele minden elemre meg kell adni a magasságot, a body-t és a html-t is beleértve.

```
body, html { height: 100%; }
```

A megoldás



https://codepen.io/gincsai/pen/PJxBGx

A fájdalmas következmény

- A %-os magasság csak akkor működik, ha
  - a szülő magassága nem `auto` vagy %,
  - vagy ha %, akkor az ő szülőjére kell igaz legyen a fenti.
- %-os magassághoz a fában az elem fölött fix magasságnak kell lennie, anélkül, hogy azt egy `auto` félbeszakítaná.
  - Ha nem szeretnénk fix magasságot, és mindenhol %-ot használnánk, egészen a html elemig fölfele mindennek kell legyen megadott magassága.
  - Nagyon nehéz teljesíteni, különösen ha komponensekben gondolkodunk, amiktől elvileg azt várnánk, hogy a szülőjüktől függetlenül működjenek.
    - pl. Angular direktívák

Margin, padding, box model

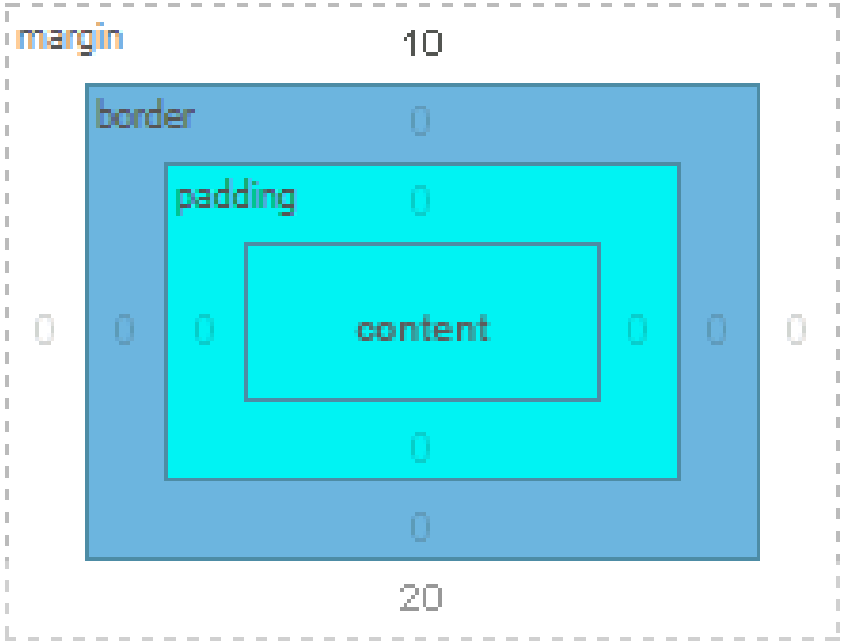
Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

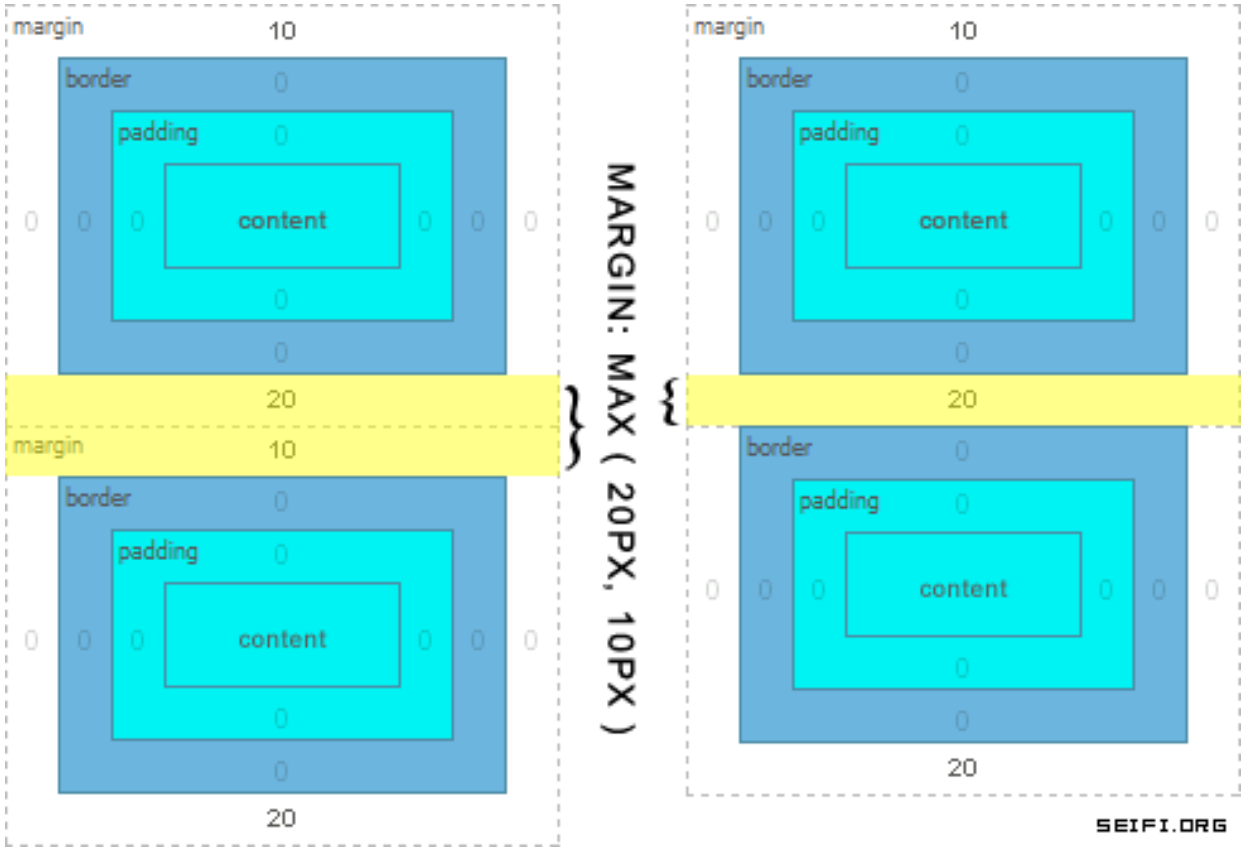
Blokk elemek – margin és padding

- A *margó* a blokk köré, a *padding* a blokk külső "körvonala" és a tartalma közé helyez térközt
- Egyszerűnek tűnik, de mindkettő váratlan mellékhatásokat tud produkálni...



A margókban sem mindig bízhatunk…

- Ha vízszintes margók találkoznak, akkor azok összeolvadnak és csak a nagyobb lesz látható.



Margin collapse

- Nem bug, feature

*„While margin collapsing is **great for written text such as paragraphs and headings etc.**, it can get somewhat tricky **if you're trying to get pixel perfect spacing between your boxes.**"*

- Ha egy blokk alsó margója találkozik az utána jövő blokk felső margójával, a kettő nem összeadódik, hanem a két érték közül a nagyobb jut érvényre.
  - Sőt, van rosszabb: az első gyerek felső margója összevonódik a szülője felső margójával

Forrás: http://www.seifi.org/css/understanding-taming-collapsing-margins-in-css.html

border-box vs content-box méretezés

- A szabvány sokáig nem egyértelműsítette, hogy minek a mérete a **`width x height`**
- A korabeli IE sokáig beleszámolta a **`border`** és **`padding`** méreteket, mivel „intuitívan” ha egy dobozra gondolunk, annak szélét is beleértjük – ez a **`border-box`** méretezés
- Később a W3C viszont úgy pontosította a definíciót, a **`width x height`** csak a tartalom mérete kell legyen, és **még a `padding` sem** számít bele – ez a **`content-box`** méretezés
- CSS3-ban megadható, hogy melyiket használja.

Részletek: http://guyroutledge.github.io/box-model/

50% széles dobozok 2-2px paddinggal

- A dobozok szélessége valójában "**50% + 2*2px**", ami együtt **több, mint 100%.**
- Mivel nem férnek el egymás mellett, ezért egymás alá töri őket a layout motor.
- Ráadásul ezt CSS2-vel nem is lehetett megoldani!
  - Csak úgy, ha 50%-ról fix (px) szélességre váltunk, amiből ki tudjuk vonni a dobozonként 4 pixelnyi paddingot!
- CSS3-ban a megoldás a calc(...)
  - **width: calc(50% - 2px)**

Támogatottság ellenőrzése CSS-ben

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Támogatottság ellenőrzése

- Fejlesztés közben nézzünk utána!
- Futásidőben "pótolni" a CSS hiányokat többnyire jóval nehezebb, mint a JavaScript API-kat.
  - Ha nincs mondjuk LocalStorage, attól az oldal alapvetően még működőképes maradhat az alkalmazás.
  - De ha a layoutot flexboxra építettük, akkor annak a hiánya teljesen új struktúrát kíván meg az oldaltól...
  - Persze ha csak animációk, díszítések maradnak le régi böngészőkben, akkor az még rendben van, ha olvasható marad a tartalom.

És nem lehetne futásidőben pótolni?

- JavaScriptből próbáljuk utánozni a működést ("polyfill")
- Egyszerűbb CSS-sel helyettesítjük
  - gradiens helyett kép
  - Az "újabb" szabálynak erősebb specificitást adunk, és akkor az a modern böngészőkben felülírja a régit, az elavultak meg figyelmen kívül hagyják.
- Erre találták ki a `@supports` blokkot

A @supports blokk

- Ha egy `propery: érték` pár támogatott, érvényre jutnak a szabályok, ha nem támogatott, akkor nem jutnak érvényre.
- Ha egy böngésző *egyáltalán nem* ismeri a `@supports` blokkot, a feltételtől függetlenül figyelmen kívül hagyja a tartalmát
- Fontos figyelni, hogy a @supports-os szabály legyen később (erősebb).

```
section { float: left}
@supports( display: flex ){
    section { display: flex; float: none}
}
```

Sajnos a támogatottság nem "bináris"

- Attól még, hogy egy böngésző "támogat" egy CSS3 feature-t, nem biztos, hogy a legújabb szintakszist, vagy az összes képességet támogatja.
- Ez egy régi probléma a CSS világában, de a CSS3-on dolgozókat nem is érte váratlanul, és igyekeztek jobban kezelni, mint korábban...

A megoldás: vendor prefixek

- Amikor az Internet Explorer implementálta a CSS **`width`** és **`height`** propertyket, pontosabb definíció híján úgy értelmezték, hogy a padding beleszámít
  - Máig rejtélyes okokból, végül nem ez került a "szabványba" – nem kis fejfájást okozva a több böngészőt is támogatni kívánó fejlesztőknek.
- Amikor a CSS3-on elkezdtek dolgozni, a böngészőgyártók megegyeztek abban, hogy minden új propertyt prefixszel látnak el, amíg nem véglegesedik a specifikációjuk, hogy elkerüljék a névütközést a később "szabványra emelkedő" verzióval.

Vendor prefix példa: flexbox

```
body {
    display: -webkit-box;
    display: -moz-box;
    display: -webkit-flex;
    display: -ms-flexbox;
    display: flex;
    -webkit-box-direction: normal;
    -moz-box-direction: normal;
    -webkit-box-orient: vertical;
    -moz-box-orient: vertical;
    -webkit-flex-direction: column;
    -ms-flex-direction: column;
    flex-direction: column;
    -webkit-box-align: stretch;
    -moz-box-align: stretch;
    -webkit-align-items: stretch;
    -ms-flex-align: stretch;
    align-items: stretch;
}
```

Mi a baj a vendor prefixekkel?

- Nagyon redundáns, könnyű elrontani
  - Általában generáljuk őket (pl. LESS autoprefixer)
- Nagyon nehéz debuggolni a viselkedést
  - Gyakran nem csak a szintaktika más…

Stílus-kaszkád

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Honnan jöhetnek a stíluslapok?

- Egy csomó helyről.
  - Böngésző alapértelmezés
  - Hivatkozás `<link>` taggel (HTML-ből külső CSS hivatkozás)
  - Hivatkozás `@import`-tal (másik CSS fájlból hivatkozva)
  - `<style>` tag (HTML-ben megadott CSS szabályok)
  - `style` attribútum (tag-re téve közvetlenül)
- Ha pedig ugyanarra az elemre több selector is illik, és a megadott szabályok közül egy vagy több ugyanarra a tulajdonságra próbál hatni, akkor valamilyen rendszer szerint el kell dönteni, melyik jusson érvényre.

Mi alapján számítódik a súlyozás?

1) Fontosság
2) Származás
3) Specifikusság
4) Forrás sorrend

1) Fontosság

- Fontos: ami !important. Minden más „nem fontos”.

```
.alert-message { color: red !important; }
```

- Főleg a debug folyamat mellékhatásaként, vagy a bootstrap alapértelmezéseivel való bunyózás közben fordulhat elő, hogy ilyesmire vetemedünk.
- Kényelmes, egyszerű, bár gyakran nem működik.
  - Pl. egy inline elemre hiába állítjuk be, hogy `height: 100% !important`, az semmit nem fog csinálni.
- De még ha működik is, nem helyes, mert karbantarthatósági problémákat okozunk vele.

Házi szabályok az !important használatára

- Alapvetően nem-nem, soha!
  - Értsük meg, melyik szabályt nem tudjuk nélküle felülírni, és írjunk "erősebb" selectort, és/vagy
  - fontoljuk meg, szükséges-e, hogy az előző selector szabálya ennyire erős legyen, és próbáljuk meg azt gyengíteni
    - LESS vagy SCSS használatakor nagyon könnyű "véletlenül" túl erős szabályokat írni!
- Ha este 10 van és holnap release...
  - Ha ez a megoldja a problémát, csináljuk, de írunk ki róla egy taszkot, hogy ezt majd valaki csinálja meg rendesen.
  - TODO komment nem ér, mert az a világ végéig ott marad.

2) Származás

- A **fontosság** és a stíluslap származási helye együtt határozzák meg a szabály prioritását / erősségét. Növekvő sorrendben:
- Böngésző alapértelmezett stíluslapja
- ~~Normál szabályok a felhasználói stíluslapban~~
- Normál szabályok a szerzői stíluslapban
- Fontos szabályok a szerzői stíluslapban
- ~~Fontos szabályok a felhasználói stíluslapban~~

3) Specifikusság

- A fontossággal ellentétben ez nem a tulajdonság-beállítás, hanem a selector szintjén dől el.
- A nagyobb „specifikusságú” selector az erősebb
  - tehát ütköző tulajdonság-beállítások esetén a specifikusabb selector által definiált érték jut érvényre
- A specifikusság egy négy helyiértékből álló számsor:
  - Inline stílus (igen: 1; nem: 0)
  - ID selectorok száma
  - class, attribútum és pseudo-class hivatkozások száma
  - elemekre és pszeudoelemekre való hivatkozások száma

Példa 0;2;4;5-ös specifikusságra

```
header#myhead ul#main-nav li
   > .sub-menu li.child
   a:not(.unimportant):hover {
      color: red;
}
```

- inline stílus: nem (0)
- ID: `#main-nav`, `#myhead` (2)
- class: `.sub-menu`, `.child`, `.unimportant`, `:hover` (4)
- elemek: `header`, `ul`, `li`, `li`, `a` (5)

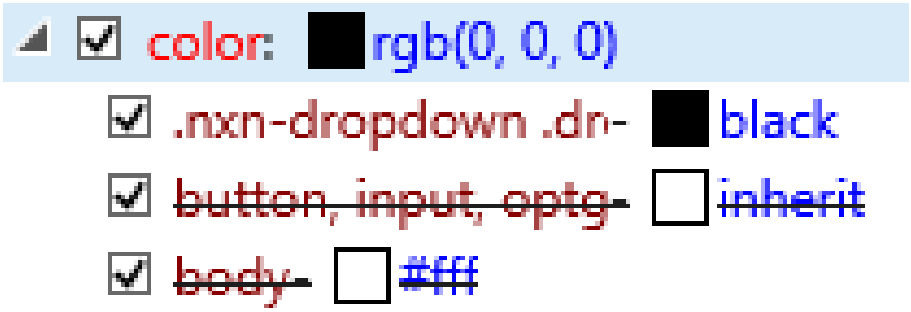
http://www.standardista.com/wp-content/uploads/2012/01/specificity3.pdf

4) Forrás sorrend

- Ha két deklarációnak
  - Azonos súlyú a forrása (fontosság, származás).
  - Azonos a specifikussága.
- ...akkor a forrás sorrend dönt.
  - Vagyis egyszerűen az, melyik stílust definiálták „lejjebb” a fájlban, vagy melyik külső fájl / `<style>` blokk lett később megadva a HTML-ben.
  - Ha `@import`-tal behivatkozunk valamit a fájl elején, azt a fájlban később következő dolgok felülírhatják.

Hogyan látjuk a szabály felülírását?

- Ha specifikusabb szabályt adunk meg
  - body-ra fehér
  - input tag-re öröklődik
  - .nxn-dropdown-ra class-al ellátott elemre fekete



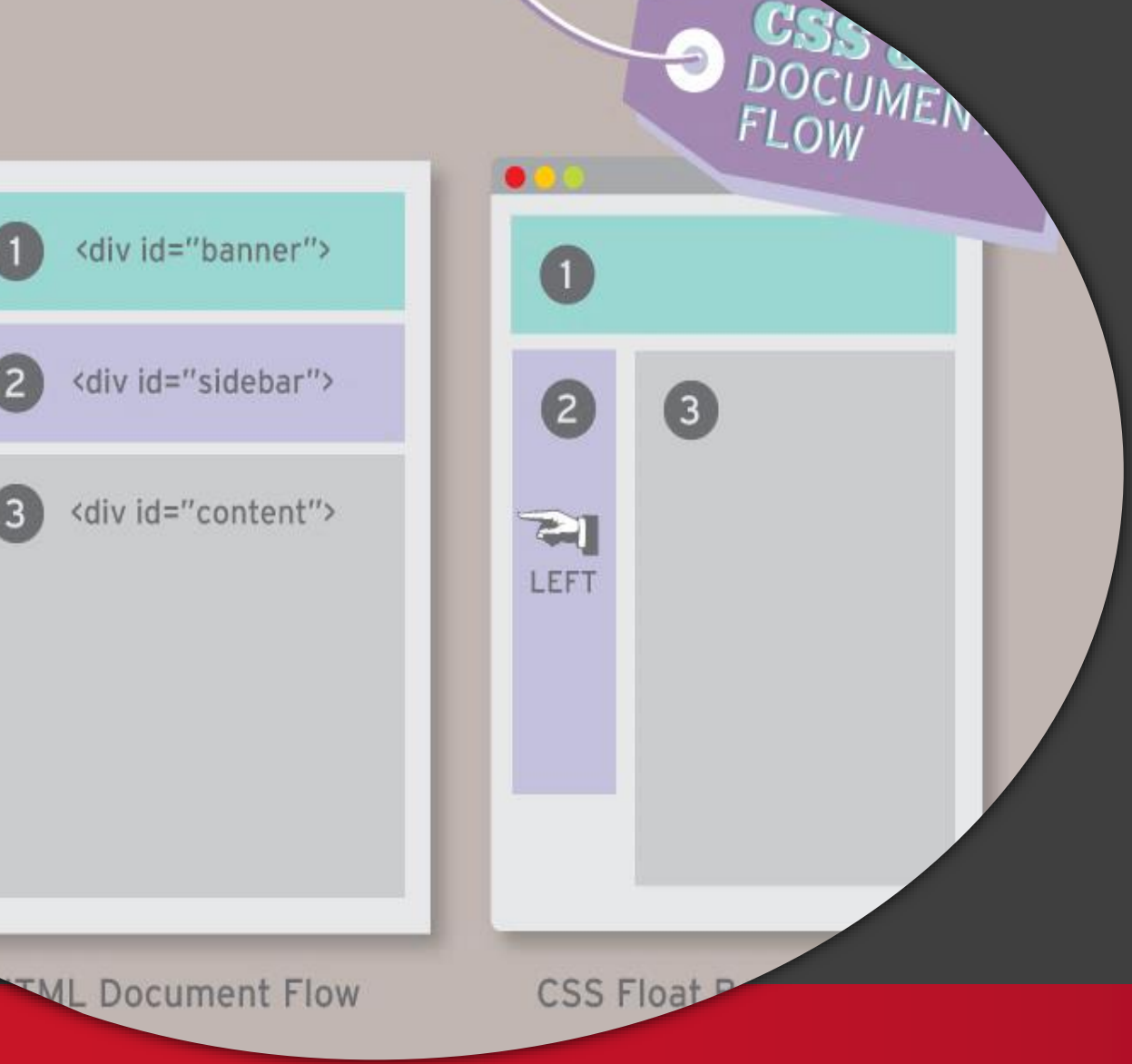
Tehát: mi alapján számítódik a súlyozás?

- Emlékeztetőül
  1) Fontosság (!imporant)
  2) Származás (böngésző alap, vagy saját CSS)
  3) Specifikusság (pl: 0;2;4;5 specifikusság)
  4) Forrás sorrend (melyiket töltöttük be később)
- Ez alapján mindig tudni fogjuk, milyen érték állítódik be egy adott tulajdonságra?


One more thing...

Öröklés

- Bizonyos tulajdonságok megörökölhetik a szülő elemen beállított értéküket.
  - Pl. `font-size`, `line-height`, `text-align` stb.
- Bizonyos tulajdonságok eleve így viselkednek, de mi is előidézhetjük az `inherit` érték beállításával.
  - Ha pedig esetleg a tulajdonság CSS ajánlás szerinti alapértelmezett értéket szeretnénk visszaállítani, az `initial` kulcsszóval tehetjük meg.
  - Bónusz: minden, színt elfogadó tulajdonságra beállítható a `currentcolor` érték, ami az aktuális szín értékét jelenti (dinamikusan)



Dokumentum folyam

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Miről van szó?

- A HTML alapvetően egy dokumentumformátum.
- Ezért a HTML-ben szereplő tartalom dokumentumszerűen próbál viselkedni.
  - balról jobbra, fentről le folyó szövegként
- Ez a „normál dokumentumfolyam”
- Webalkalmazások készítésekor azonban gyakran kell dolgokat egymás mellé, vagy „Z irányban” egymás fölé tenni.
- Ezek egyrészt kitörtnek a folyamból, másrészt befolyásolják a folyam működését.

A normál dokumentumfolyam

- Inline folyam
  - elemek egymás mellett, a szövegiránynak megfelelően
  - magasságukat, szélességüket a tartalmuk mérete diktálja
- Blokk folyam
  - elemek egymás alatt, gyakorlatilag új sort kezdve
  - a tartalom csak a magasságukat diktálja, szélességben kitöltik a szülő teljes szélességét
    - a tartalom lehet szöveg de más blokk elem(ek) is
  - az elsődleges layout formáló eszközeink a blokk elemek
    - nem véletlenül írunk annyi -et a kódba…

A blokk folyam viselkedése

- Mivel blokk elemek egymás alatt helyezkednek el, a **box-modell tulajdonságok** változtatása (margó, magasság stb.)
  - > **nem befolyásolja** az előttük lévő elemek elhelyezkedését
  - > **szinte mindig befolyásolja** az utánuk jövő elemek elhelyezkedését
- Triviálisnak tűnik, de amikor arról van szó, hogyan tudja egy sornyi CSS „elrontani” a 300 sorral lejjebb definiált dolog megjelenését, akkor pontosan az ilyen összefüggésekről van szó.

Block folyam vs függőleges pozícionálás

A PEN BY Gincsai Gábor

Save | Fork | Settings | Change View

HTML

```
<div class="wrapper">
  <div class="box"></div>
  <div class="box box-special"></div>
  <div class="box"></div>
</div>
```

CSS

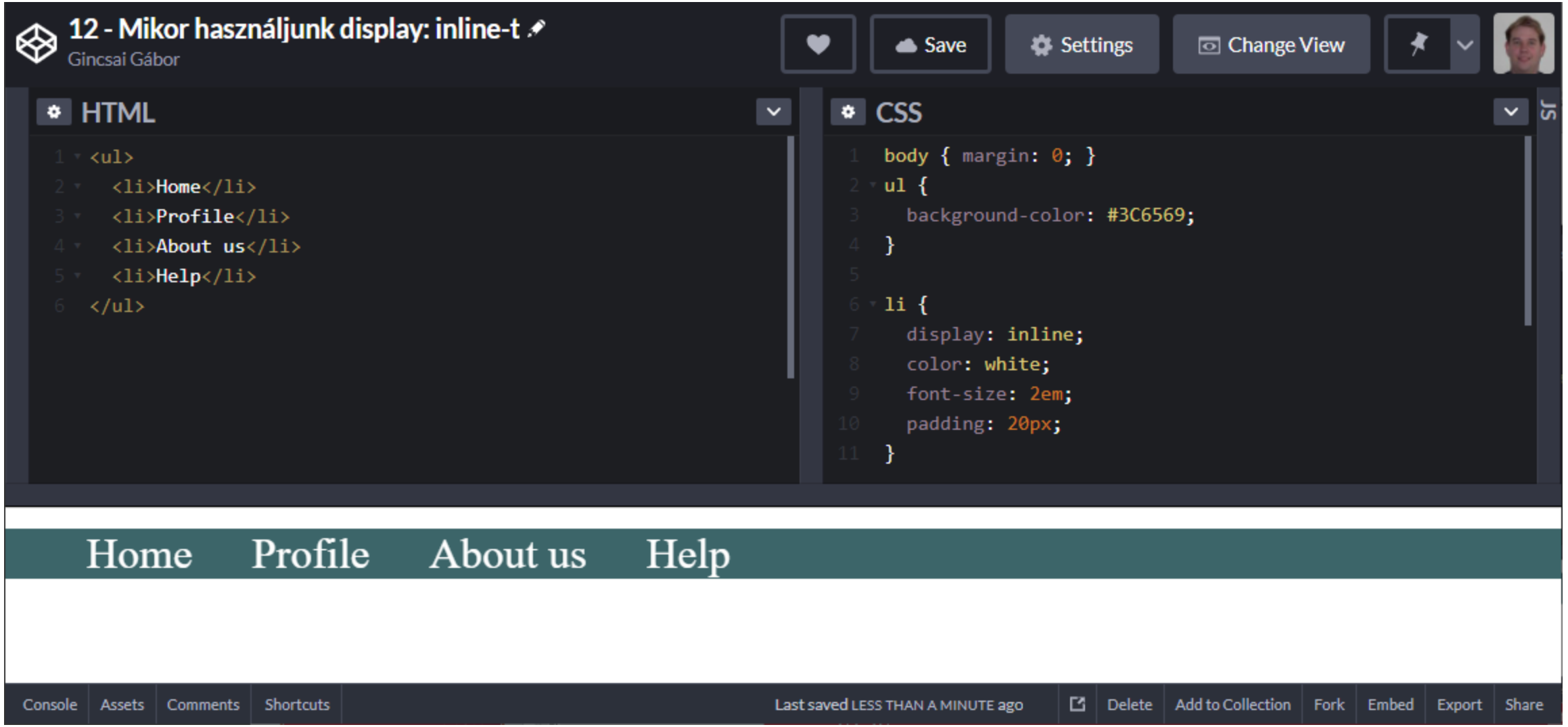
```
body {
  background-color: #1d1f20;
}
.wrapper {
  box-sizing: border-box;
  color: #dfdfdf;
  border: 20px solid currentcolor;
  background-color: currentcolor;
  height: 180px;
}
.box {
  height:40px;
  background:#cc3f85;
  margin:0 0 10px;
}
.box-special {
  background:#953fcc;
  height: 90px;  /* az alatta lévő fog kilógni */
}
```

https://codepen.io/gincsai/pen/OxrVVm

A display property

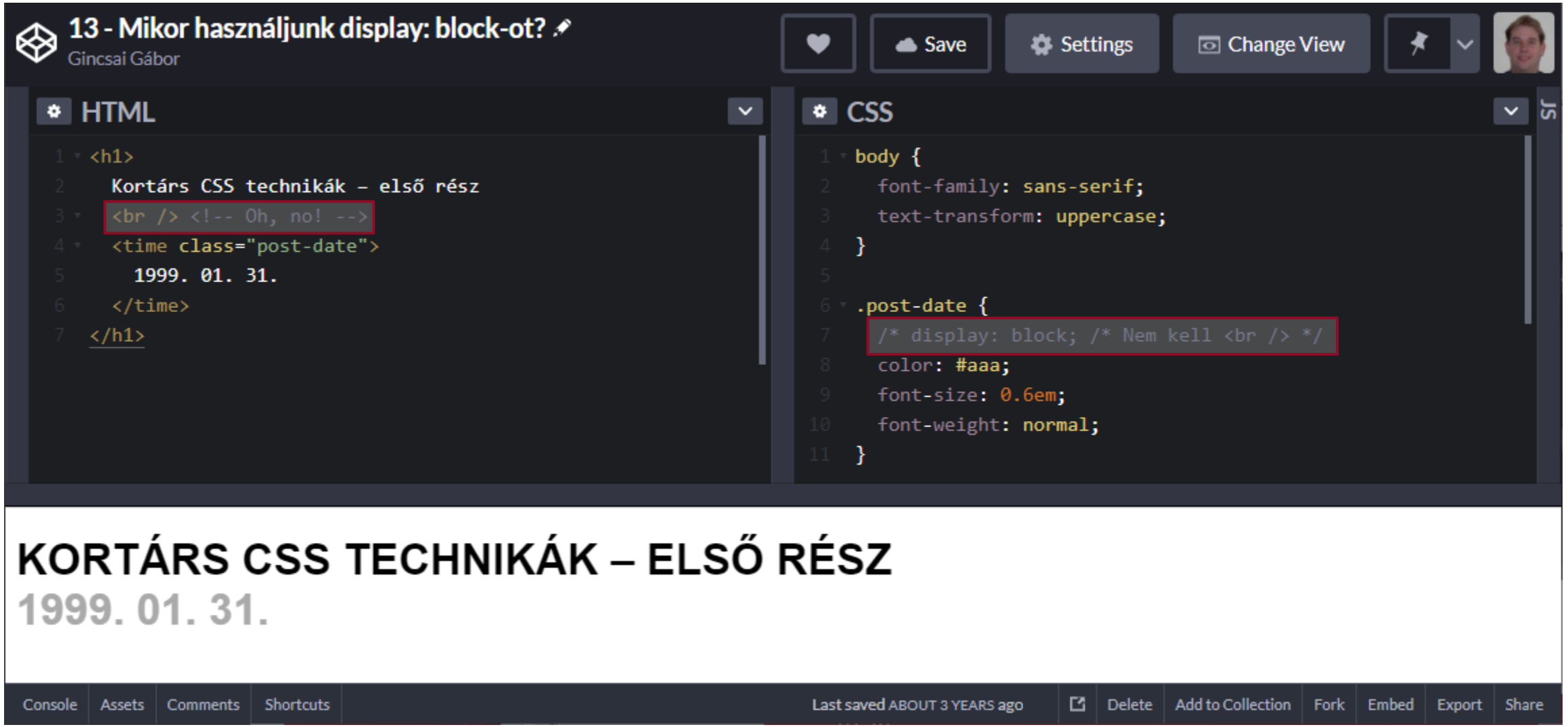
- Az elemek folyambeli viselkedését a már említett `display: block | inline` beállításával tudjuk testre szabni.
- Meglepően sok lehetséges értéke van:
  - a közismert **none** és az `inline-block`
  - a `<table>` nélküli táblázatokhoz használható `table-column`, `table-cell` stb.
  - a `flex`, amiről még lesz szó
  - a `list-item` is önálló típus
- A legnagyobb kényelmetlenség, hogy a **none** is a **display-en** állítható be
  - nagyon megnehezíti robusztus "**show-hide**" logikák megírását.
- A *hide* még oké, de a *show* milyen display értéket állítson be?
  - Nem véletlen, hogy Bootstrapben is van külön **visible-xs-block** és **visible-xs-inline** stb., de rejtésre csak a hidden-xs van.

Mikor jó a display: inline?



https://codepen.io/gincsai/pen/qPLdrd

Mikor jó a display: block?



https://codepen.io/gincsai/pen/OxrVOO

Dokumentum folyam megtörése:
position, z-index és a float

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

A normál folyam megtörése pozicionálással

- A `position` tulajdonság átállításával kivehetjük az elemet a normál folyamból, és általunk megadott koordináták mentén helyezhetjük el.
  - `position` deklarációt általában kombináljuk a `top`, `right`, `bottom`, `left` és `z-index` tulajdonságokkal.
- Értékei `static`, `relative`, `absolute`, `fixed`
  - Ezek abban különböznek egymástól, hogy mihez képest értelmezzük a koordinátákat, és hogy a környezet hogyan reagál az elem kiszakítására a folyamból.

position: static

- Ez az alapértelmezés: az elem a normál dokumentumfolyam része
- A `top`, `right`, `bottom`, `left` és `z-index` tulajdonságok hatástalanok.
- „Kézzel” ritkán állítjuk be, legfeljebb ha felül akarunk írni egy korábbi átállítást.

position: relative

- Az elem eltolása a normál folyam szerinti helyéhez képest
- Az eredetileg „neki szánt” hely üresen marad, a többi elem elhelyezkedése így nem változik
  - Ha nem figyelünk oda, az üres hely is hülyén fog kinézni, és még takarás is lesz
- Az elem új rétegre kerül, a normál folyambeli elemek fölé
  - Az elemek z-sorrendjének variálására nagyon sokszor `position: relative` a megoldás, és nem a `z-index`!
- Az új réteghez új koordinátarendszer is dukál
  - A gyerekelemek abszolút pozicionálása *ehhez az elemhez képest* lesz abszolút ☺

14 - position: relative

Gincsai Gábor

Save | Settings | Change View

HTML

```
<main>
  <p>Bacon ipsum dolor amet pastrami boudin ribeye, andouille strip steak shank kevin bacon shankle.</p>
  <figure>
    <img src="http://baconmockup.com/200/150" />
  </figure>
  <p>Hamburger ham hock pancetta bresaola cupim, filet mignon short ribs cow turducken picanha salami. Beef cow hamburger pork chop jowl. Meatloaf tri-tip sirloin ham shank andouille prosciutto hamburger chuck.</p>
  <p>Picanha pork chop shank, meatloaf ground round short ribs beef ribs short loin venison porchetta fatback strip steak tongue drumstick.</p>
</main>
```

CSS

```
main{ width: 40%; }
figure {
  float: left;
  margin: 5px;
  padding: 15px;
  background: pink;
}
img {
  position: relative;
  left: 50px;
}
```

Bacon ipsum dolor amet pastrami boudin ribeye, andouille strip steak shank kevin bacon shankle.

Hamburger ham hock pancetta bresaola
m, filet mignon short ribs cow
cken picanha salami. Beef cow
burger pork chop jowl. Meatloaf tri-
rloin ham shank andouille prosciutto
burger chuck.

nha pork chop shank, meatloaf
nd round short ribs beef ribs short
venison porchetta fatback strip steak
tongue drumstick.

Console | Assets | Comments | Shortcuts | Last saved LESS THAN A MINUTE ago | Delete | Add to Collection | Fork | Embed | Export | Share

https://codepen.io/gincsai/pen/BwvmVE

15 - position: relative z-index

Gincsai Gábor

Save | Settings | Change View

HTML

```
<main>
<p>Bacon ipsum dolor amet pastrami boudin ribeye, andouille strip steak shank kevin bacon shankle.</p>
<figure>
  <img class="first" src="http://baconmockup.com/100/150" />
</figure>
<figure>
  <img class="second" src="http://baconmockup.com/120/150" />
</figure>
<p>Hamburger ham hock pancetta bresaola cupim, filet mignon short ribs cow turducken picanha salami. Beef cow hamburger pork chop jowl. Meatloaf tri-tip sirloin ham shank andouille prosciutto hamburger chuck.</p>
<p>Picanha pork chop shank, meatloaf ground round short ribs beef ribs short loin venison porchetta fatback strip steak tongue drumstick.</p>
</main>
```

CSS

```
  padding: 15px;
  background: pink;
}
img.first {
  position: relative;
  left: 70px;
  top: 10px;
}
img.second {
  z-index: 9999;
  /*position: relative;*/
}
```

Bacon ipsum dolor amet pastrami boudin ribeye, andouille strip steak shank kevin bacon shankle.

Hamburger ham hock pancetta bresaola cupim, filet mignon short ribs cow turducken picanha salami. Beef cow hamburger pork chop jowl. Meatloaf tri-tip sirloin ham shank andouille prosciutto hamburger chuck.

Picanha pork chop shank, meatloaf ground round short

Console | Assets | Comments | Shortcuts | Last saved LESS THAN A MINUTE ago | Delete | Add to Collection | Fork | Embed | Export | Share

https://codepen.io/gincsai/pen/QqzOzw

position: relative – mikor használjuk?

- Ha valakitől béna assetet kaptunk, és szeretnénk épp csak 1-2 pixellel arrébb tolni valamit anélkül, hogy bármi más változna az oldalon.
  - Ha úgy szeretnénk 1-2 pixellel arrébb tolni, hogy a környezet is arrébb másszon, arra használjunk margót
- Ha szeretnénk egy normál folyambeli elemet normál folyamon kívüli elem fölé helyezni anélkül, hogy a helyét megváltoztatnánk.
- Ha a gyerekelemeket abszolút pozícionálni szeretnénk, de nem a dokumentumhoz, hanem ehhez az elemhez képest.

position: absolute

- Az elem tetejének, bal oldalának stb. precíz elhelyezése az aktuális koordinátarendszerben.
  - Aktuális koordinátarendszer: a legközelebbi szülő, ami `position: relative`, ha nincs ilyen, akkor a `<body>`
- Teljesen kikerül a normál folyamból, a többi elemet úgy rendezi a böngésző, mintha ő nem is létezne.
  - Ezzel pl. a szülő elem méretezését teljesen össze lehet zavarni.
- Új rétegre kerül (úgy, hogy nem marad alatta üres hely)
  - Kombinálva az előzővel ez általában takarást jelent, ha nem ésszel használjuk.

16 - position: absolute

Gincsai Gábor

Save | Settings | Change View

HTML

```
<main>
<p>Bacon ipsum dolor amet pastrami boudin ribeye, andouille strip steak shank kevin bacon shankle.</p>
<figure>
  <img src="http://baconmockup.com/200/150/" />
</figure>
<p>Hamburger ham hock pancetta bresaola cupim, filet mignon short ribs cow turducken picanha salami. Beef cow hamburger pork chop jowl. Meatloaf tri-tip sirloin ham shank andouille prosciutto hamburger chuck.</p>
<p>Picanha pork chop shank, meatloaf ground round short ribs beef ribs short loin venison porchetta fatback strip steak tongue drumstick.</p>
</main>
```

CSS

```
main{
  width: 50%;
}
figure {
  float: left;
  margin: 5px;
  padding: 15px;
  background: pink;
  /*position: relative;*/
}
img {
  position: absolute;
  /* top: 0; /* NEM alapértelmezés a 0! */
}
```

Bacon ipsum dolor amet pastrami boudin ribeye, andouille strip steak shank kevin bacon shankle.

Hamburger ham hock pancetta bresaola cupim, filet mignon short ribs cow turducken
amburger pork chop jowl. Meatloaf tri-tip sirloin ham shank
ger chuck.

Pic f ground round short ribs beef ribs short loin venison porchetta
fat ck.

Console | Assets | Comments | Shortcuts | CodePen: Unlock all of CodePen | Last saved ALMOST 3 YEARS ago | Delete | Add to Collection | Fork | Embed | Export | Share

position: absolute (folyt.)

- Blokkszerű viselkedést vesz fel: `width`, `height` stb. tulajdonságokra reagál.
  - > De a blokk elemekkel ellentétben nem tölti ki a szülő szélességét, ehelyett a gyerekelemek számára minimálisan szükséges méretet veszi fel, ha mást nem mondunk.
- Az abszolút pozícionált elemeken ezért gyakran kézi `width` és/vagy `height` állítást is végzünk.

position: absolute – előnyök / hátrányok

- Fő előny: full control
  - A sitebuilderek elsődleges kedvenc eszköze a '90-es évek végén: mindenről megmondhatjuk, hogy hol van, és senki nem lökhet félre senkit.
  - De ez csak fix dokumentumméretre működik jól…
- Fő hátrány: rugalmatlan
  - Főleg alapértelmezésben, ha a <body> koordinátarendszeréhez képest használjuk: ha egy dolgot arrébb tolunk, a többi elemet is „kézzel” kell arrébb toljuk az útjából.
  - Nekünk kell figyelni, hogy ne takarja ki a normál flow elemeit (margó vagy padding használatával)

positon: absolute – mikor használjuk?

- Kisebb komponenseken belül használjuk
  - Több rétegre van szükségünk vagy pontos elhelyezésre
  - De nincs sok más elem, amire figyelnünk kell menet közben.
- Ha gondoskodunk róla, hogy a komponens gyökere helyesen működjön a normál folyam részeként (nem lóg ki belőle semmi), és arra alapozzuk a koordinátarendszert, többnyire megússzuk a karbantartási problémákat .
  - Plusz, ha ügyesen használjuk, nem is *annyira* rugalmatlan

17 - position: absolute layering

Gincsai Gábor

Pen saved.

Save | Settings | Change View

HTML

```
<figure>
  <img src="http://baconmockup.com/300/200/" /><figcaption>Delicious meat</figcaption>
</figure>
<figure>
  <img src="http://baconmockup.com/299/200/" /><figcaption>Real food</figcaption>
</figure>
```

CSS

```
figure {
  position: relative;
  width: 300px;
}
figcaption {
  text-transform: uppercase;
  color: white;
  background: linear-gradient(to bottom, rgba(0,0,0,0.8) 10%,rgba(0,0,0,0) 70%);
  padding-top: 10px;
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  text-align: center;
}
```

JS

Console | Assets | Comments | Shortcuts | Last saved LESS THAN A MINUTE ago | Delete | Add to Collection | Fork | Embed | Export | Share

18 - position: absolute layering (inline-block)

Gincsai Gábor

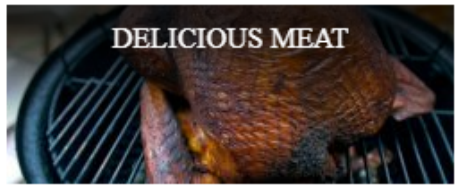
Save | Settings | Change View

HTML

```
<figure>
  <img src="http://baconmockup.com/250/100/" />
  <figcaption>Delicious meat</figcaption>
</figure>
<figure>
  <img src="http://baconmockup.com/200/150/" />
  <figcaption>Real food</figcaption>
</figure>
```

CSS

```
figure {
  position: relative;
  display: inline-block;
}
figcaption {
  text-transform: uppercase;
  color: white;
  background: linear-gradient(to bottom, rgba(0,0,0,0.8) 10%,rgba(0,0,0,0) 70%);
  padding-top: 10px;
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  text-align: center;
}
```



Console | Assets | Comments | Shortcuts | Last saved LESS THAN A MINUTE ago | Delete | Add to Collection | Fork | Embed | Export | Share

position: fixed

- Mint az `absolute`, de itt az elem akkor sem mozog az oldallal, ha elgörgetik.
- A `top`, `left` stb. koordinátarendszere: a legközelebbi szülő, amin a `transform` bármire be van állítva, ha nincs ilyen, akkor a böngészőablak belső széle.
  - Tehát a `position: relative` itt nincs hatással a koordinátarendszerre!
- Ha a szülőt nem akarjuk igazából transzformálni, a `transform: translate(0);` népszerű választás a koordinátarendszer áthelyezésére.
- De igazából sok értelme nincs.

z-index

- `position: static`-tól eltérő beállítás esetén minden elem új rétegre kerül.
- Ezek sorrendjét kontrollálja a `z-index`.
  - A nagyobb értékű elem kitakarja a kisebb értékűt.
  - Ha valakit nagyon hátra akarunk küldeni, negatív értéket is megadhatunk.
  - Alapértelmezés: `auto`
  - `auto` (vagy azonos érték) esetén a HTML-ben később következő elem kitakarja a korábban szereplőt.

20 - z-index

Gincsai Gábor

Save | Settings | Change View

HTML

```
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
```

CSS (Less)

```
div {
  width: 100px;
  height: 100px;
  position: absolute;
}
/*
div:nth-child(1) {
  z-index: 1;
}
div:nth-child(2) {
  z-index: 1;
}
div:nth-child(5) {
  z-index: -1;
}
div:nth-child(6) {
  z-index: -1;
```

JS

Console | Assets | Comments | Shortcuts

Last saved LESS THAN A MINUTE ago | Delete | Add to Collection | Fork | Embed | Export | Share

Stacking context

- A z-index egyszerű is lehetne, de nem az
  - Az ok: nem globális számról van szó, a szülők (és egyéb felmenők) értékei is számítanak!
- Pontosabban: a z-index mindig az aktuális stacking contexten belül érvényesül, így „globálisan” nem kerülhet alacsonyabb vagy magasabb rétegre, mint a stacking context gyökere
- Új stacking context gyökeret hoz létre:
  - z-index beállítása relatív vagy abszolút pozicionált, vagy flexboxban szereplő elemen
  - 1-nél kisebb opacity beállítása
  - bármilyen transform érték, ami nem none

21 - z-index container

Gincsai Gábor

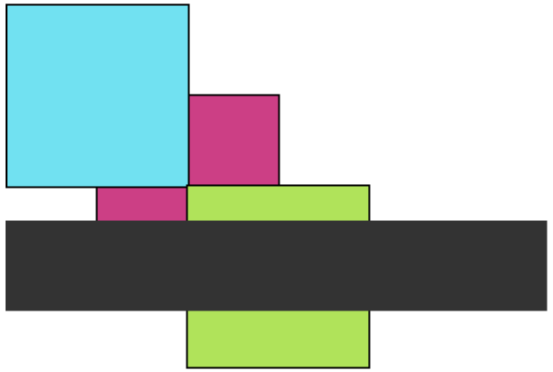
Save | Settings | Change View

HTML

```
<div class="container">
  <div class="box pink"></div>
  <div class="box blue"></div>
  <div class="box green"></div>
</div>
<section></section>
```

CSS (Less)

```
.container {
  position:relative;
  z-index:3;  /* minden benne lévő 3-asak */
}
section {
  position:relative;
  top:120px;
  z-index:5;     /* ez van felül, container: 3 ez 5 */
  background:#333;
  width:300px;
  height:50px;
}
.pink {
  z-index:1;  /* containeren belüli sorrend */
  top:50px;
  left:50px;
```



JS

Console | Assets | Comments | Shortcuts | Last saved LESS THAN A MINUTE ago | Delete | Add to Collection | Fork | Embed | Export | Share

A normál folyam megtörése – float-tal

- Eredeti célja: kép (vagy más, dobozszerű tartalom) körbefolyatása jobbról vagy balról
  - Ahogy azt egy szép dokumentumban szokás
- Elsősorban arra lett kitalálva, hogy *szöveg* folyassa körbe, de gyakorlatilag minden más tartalom *meg fogja próbálni*
  - Gyakorlatilag ez a tulajdonsága teszi lehetővé, hogy sokkal többre használjunk, mint amire eredetileg kitalálták
  - …de ez okozza a legtöbb kihívást is

22 - float image

Gincsai Gábor

Save | Settings | Change View

HTML

```
<article>
  <header>
    <h1>Dyin' ain't much of a livin', boy</h1>
  </header>
  <main>
    <img src="http://placehold.it/150x150" />
    <p> I don't think they tried to market it to the billionaire, spelunking, BASE-jumping crowd. Well, what is it today? More spelunking? What you have to ask yourself is, do I feel lucky. Well do ya' punk? The man likes to play chess; let's get him some rocks. Bruce... I'm God. I did the same thing to Gandhi, he didn't eat for three weeks. Cities fall but they are rebuilt. Heroes die but they are remembered. No, this is Mount Everest. You should flip on the Discovery Channel from time to time. But I guess you can't now, being dead and all. Multiply your anger by about a hundred, Kate, that's how much he thinks he loves you. Circumstances have taught me that a man's ethics are the only possessions he will take beyond the grave. Boxing is about respect. Getting it for yourself, and taking it away from the other guy. Dyin' ain't much of a livin', boy.
    </p>
```

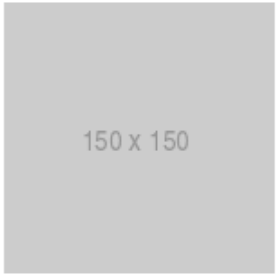
CSS (Less)

```
img {
  float: left;
  margin: 5px;
}
```

JS

Console | Assets | Comments | Shortcuts | Last saved LESS THAN A MINUTE ago | Delete | Add to Collection | Fork | Embed | Export | Share

Dyin' ain't much of a livin', boy

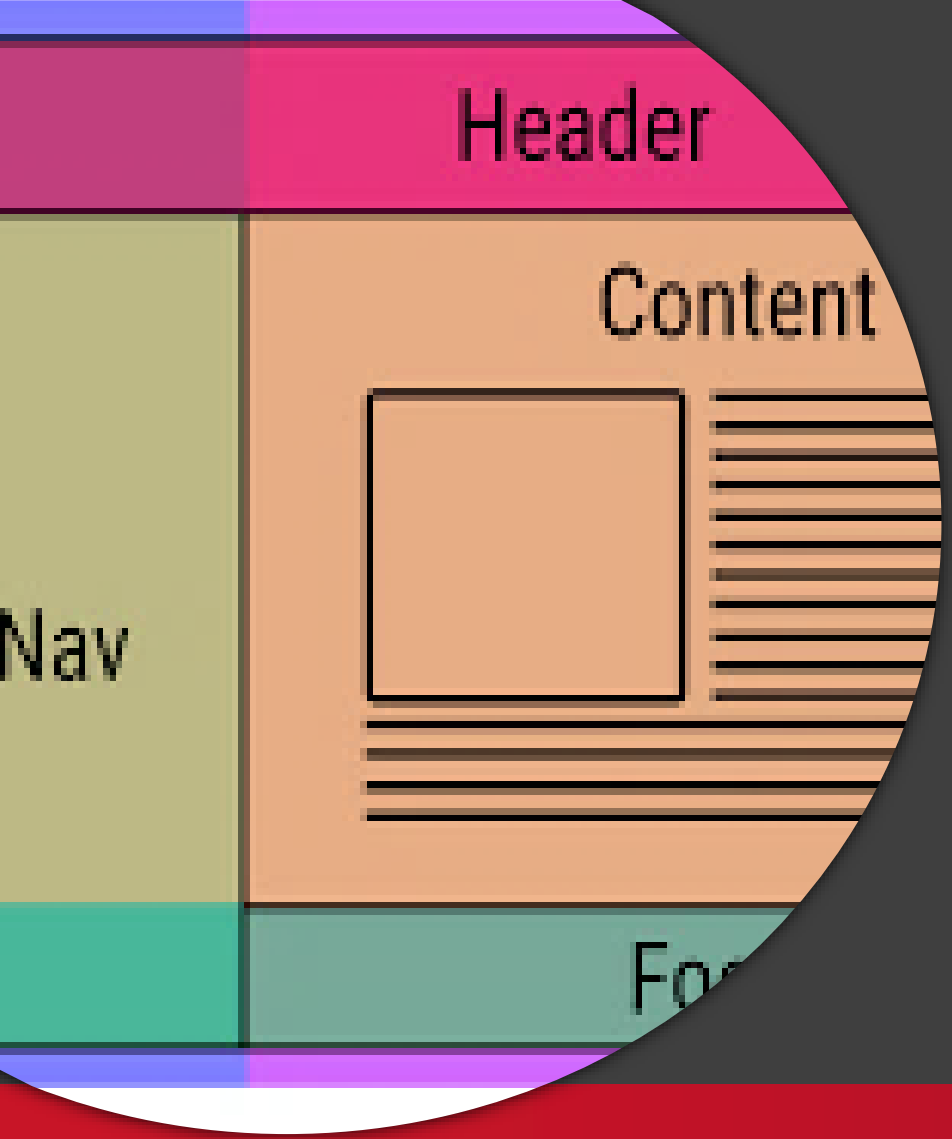


I don't think they tried to market it to the billionaire, spelunking, BASE-jumping crowd. Well, what is it today? More spelunking? What you have to ask yourself is, do I feel lucky. Well do ya' punk? The man likes to play chess; let's get him some rocks. Bruce... I'm God. I did the same thing to Gandhi, he didn't eat for three weeks. Cities fall but they are rebuilt. Heroes die but they are remembered. No, this is Mount Everest. You should flip on the Discovery Channel from time to time. But I guess you can't now, being dead and all. Multiply your anger by about a hundred, Kate, that's how much he thinks he loves you. Circumstances have taught me that a man's ethics are the only possessions he will take beyond the grave. Boxing is about respect. Getting it for yourself, and taking it away from the other guy. Dyin' ain't much of a livin', boy.

Cities fall but they are rebuilt. Heroes die but they are remembered. Ever notice how sometimes you come across somebody you shouldn't have F**ked with? Well, I'm that guy. That tall drink of water with the silver spoon up his ass. It only took me six days. Same time it took the Lord to make the world. You see, in this world there's two kinds of people, my friend: Those with loaded guns and those who dig. You dig. Well, do you have anything to say for yourself? You measure yourself by the people who measure themselves by you. No, this is Mount Everest. You should flip on the Discovery Channel from time to time. But I guess you can't now, being dead and all. You want a guarantee, buy a toaster. I now issue a new commandment: Thou shalt do the dance. This is my gun, Clyde! I don't think they tried to market it to the billionaire, spelunking, BASE-jumping crowd.

A float hatása a dokumentumfolyamra

- Első ránézésre a legrosszabb kombináció: a folyamból is kikerül és befolyásolja is azt.
  - A környező tartalom megpróbálja körbefolyni.
- Az elem *blokk-szerűvé* válik (magasság, margó stb.)
  - A border és margó természetesen befolyásolja a körbefolyó tartalom elhelyezkedését…
  - Margin collapse sosem történik float-olt elemeknél.
- Bár a folyamból kikerülnek, nem kapnak saját réteget
  - A `z-index` hatástalan!
  - Kivéve persze, ha a `position` tulajdonsággal is babrálunk.



Hagyományos layout megoldások

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

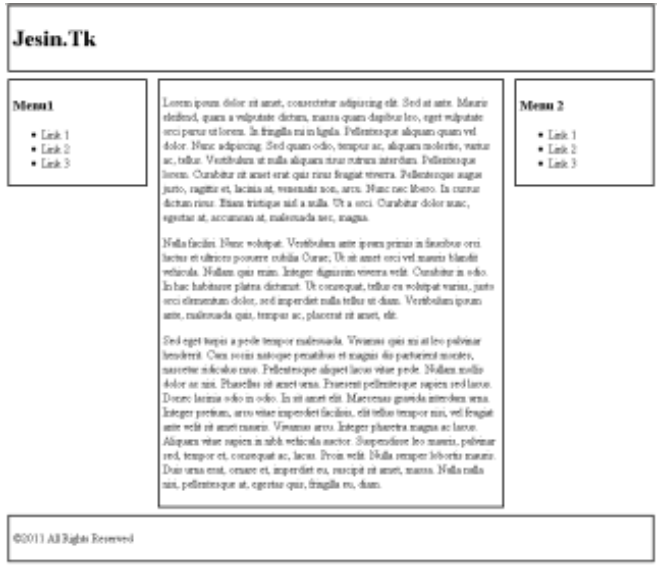
Gincsai Gábor
gincsai@aut.bme.hu

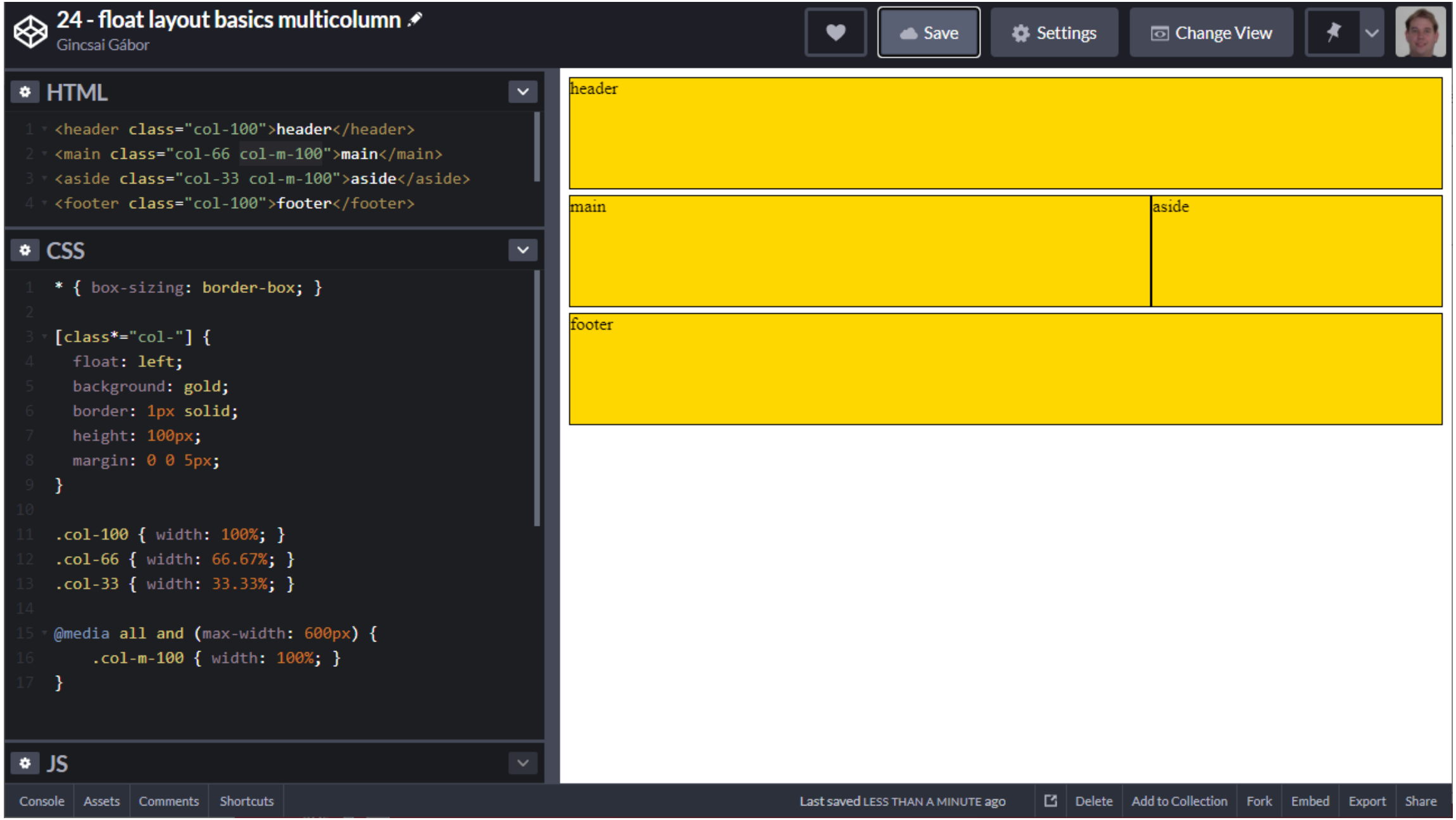
A float használata layoutra

- Layoutra a `<table>` alapú layout idejétmúlttá kikiáltását követően kezdték el használni
  - Ez "hajtja" pl. a Bootstrap 3-at is, ami jelenleg még igen elterjedt.
  - A Bootstrap 4-ben már használhatjuk az új flexbox alapú megoldást, ami sok szenvedéstől kímél meg minket.

Float blokkok halmozódása

- Ha két elemet is ugyanabba az irányba „float-olunk”, *egymás mellé* kerülnek.
  - Egészen addig, amíg vízszintesen kiférnek, utána *sortörés* következik be
    - **VIGYÁZAT:** Az új sor nem feltétlenül kezdődik a képernyő szélén
  - Mindezt összetett layoutok építésére tudjuk használni.

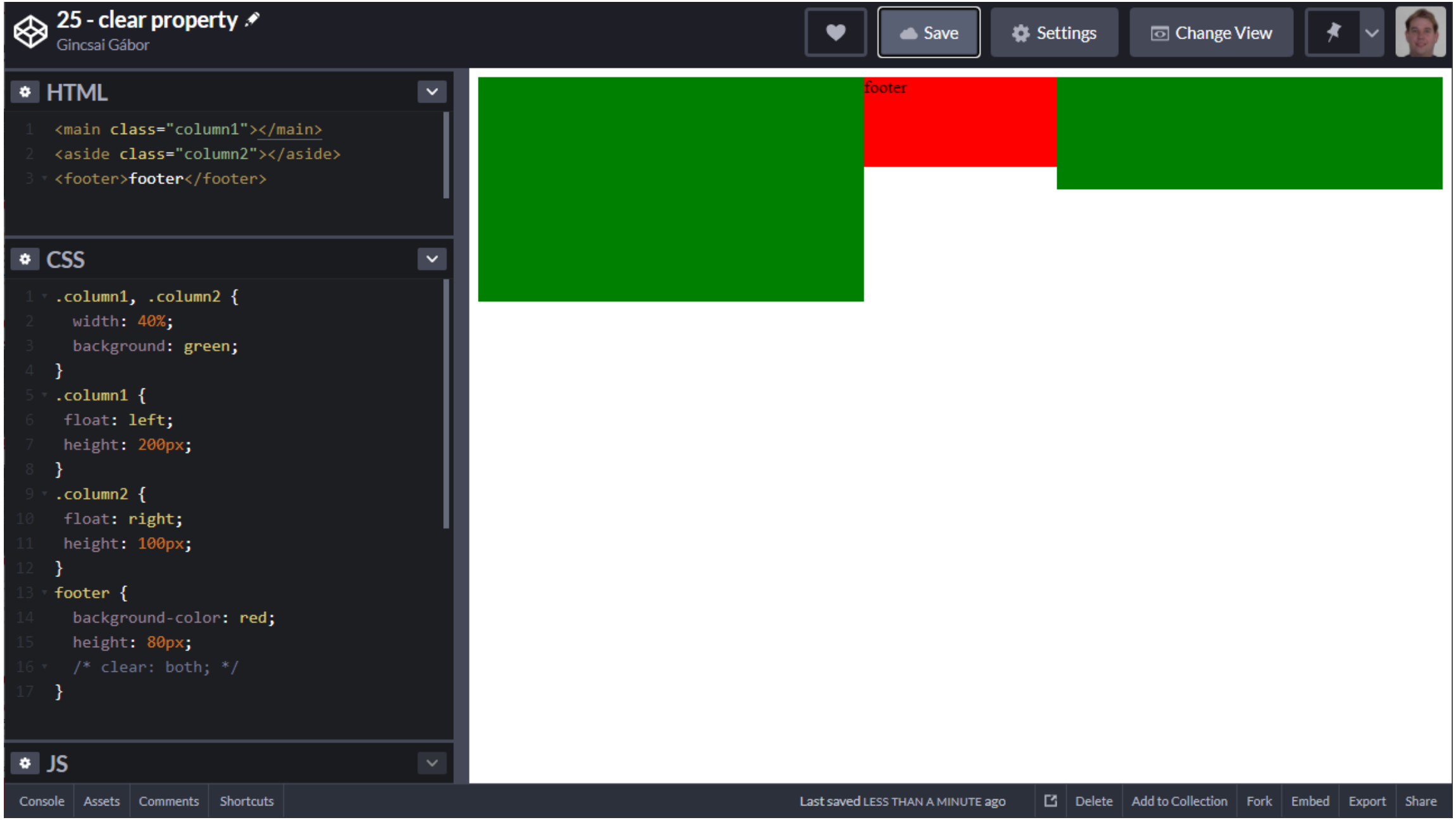


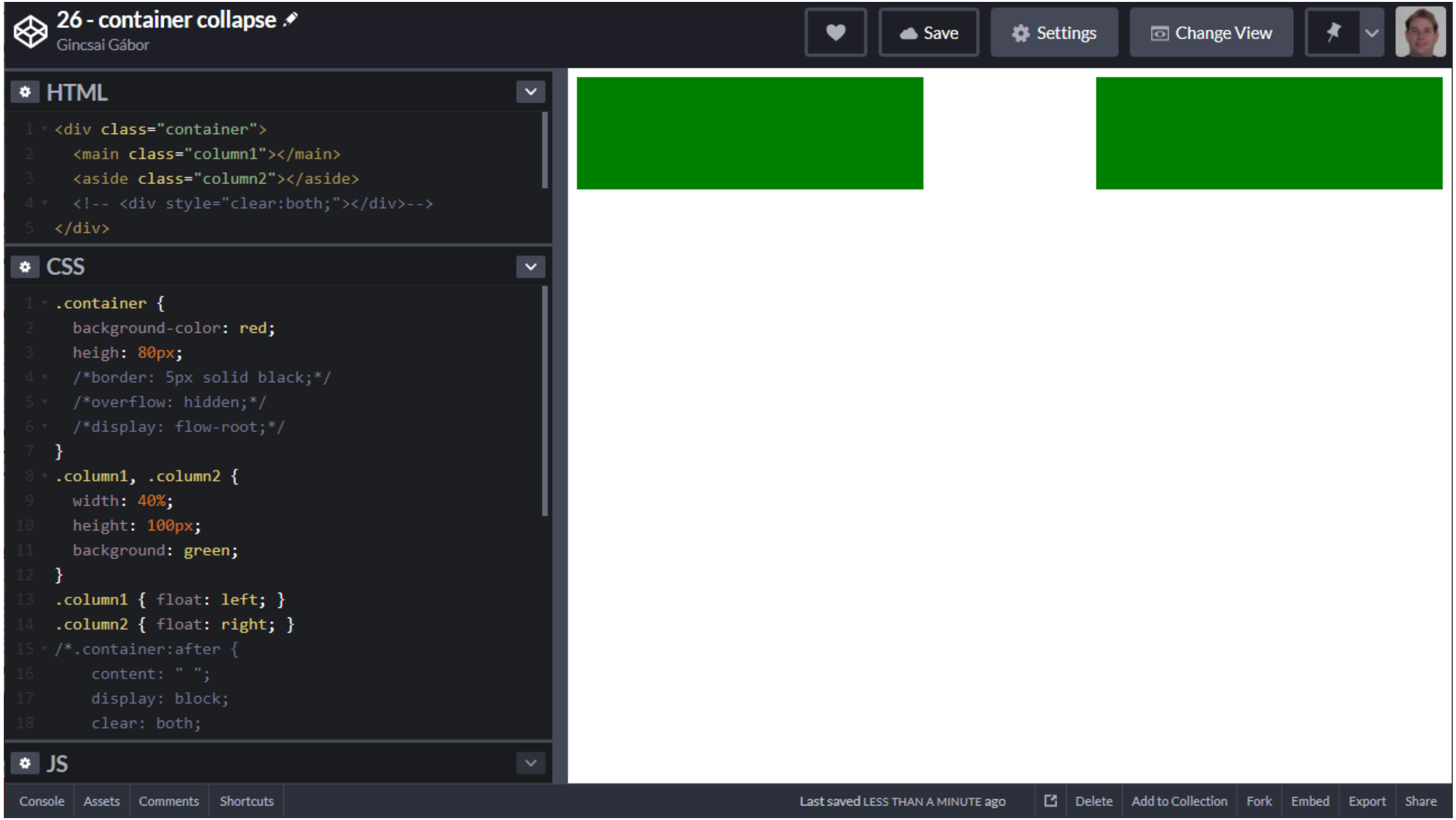


https://codepen.io/gincsai/pen/yzGxxz

Clear property

- A float arra való, hogy körbefuttathatóvá tegyen dolgokat.
  - A nem float-olt elemek szépen körbe is futják, ha módjuk van rá.
  - De így például a footer nem a lap alján van, hanem „beszorul” a két panel közé.
- Ennek megoldására való a `clear` property
  - Letiltja a körbefuttatást egyik, másik vagy mindkét oldalon.


25 - clear property
Gincsai Gábor
Save
Settings
Change View
HTML
<main class="column1"></main>
<aside class="column2"></aside>
<footer>footer</footer>
CSS
.column1, .column2 {
width: 40%;
background: green;
}
.column1 {
float: left;
height: 200px;
}
.column2 {
float: right;
height: 100px;
}
footer {
background-color: red;
height: 80px;
/* clear: both; */
}
JS
footer
Console
Assets
Comments
Shortcuts
Last saved LESS THAN A MINUTE ago
Delete
Add to Collection
Fork
Embed
Export
Share



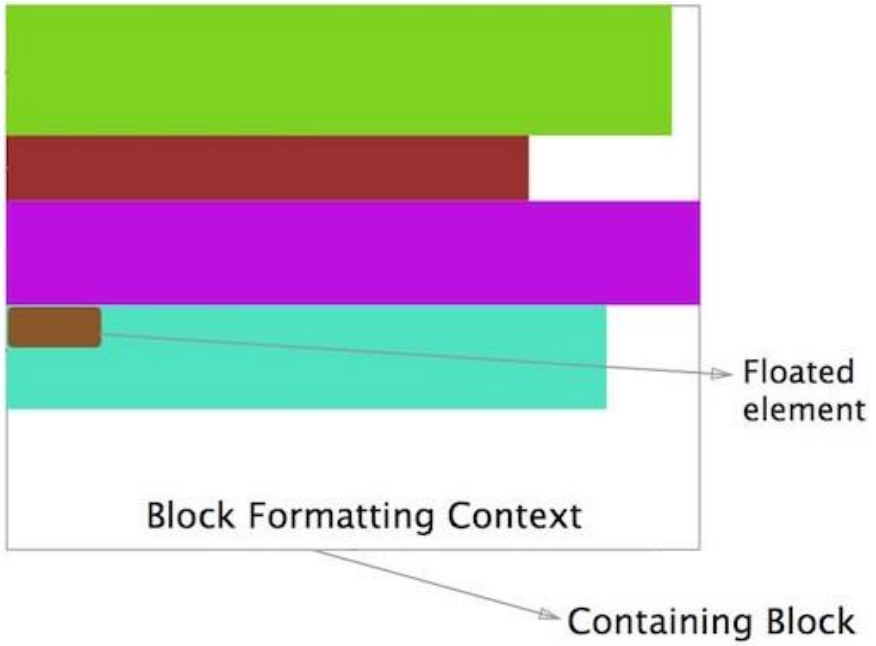
https://codepen.io/gincsai/pen/JrwBBy

Kitérő: block formatting context

- Miért volt megoldás az `overflow: hidden;`?
  - Egyáltalán: mi okozta a problémát?
- Hogy megértsük, meg kell ismerkednünk a block formatting context fogalmával.

Block formatting context

- Egy doboz, amin belül a szövegfolyam, beleértve a float-olt elemeket is, elrendezésre kerül.
- Másra is hat, pl. különálló block formatting contextek között margin collapse sincs.

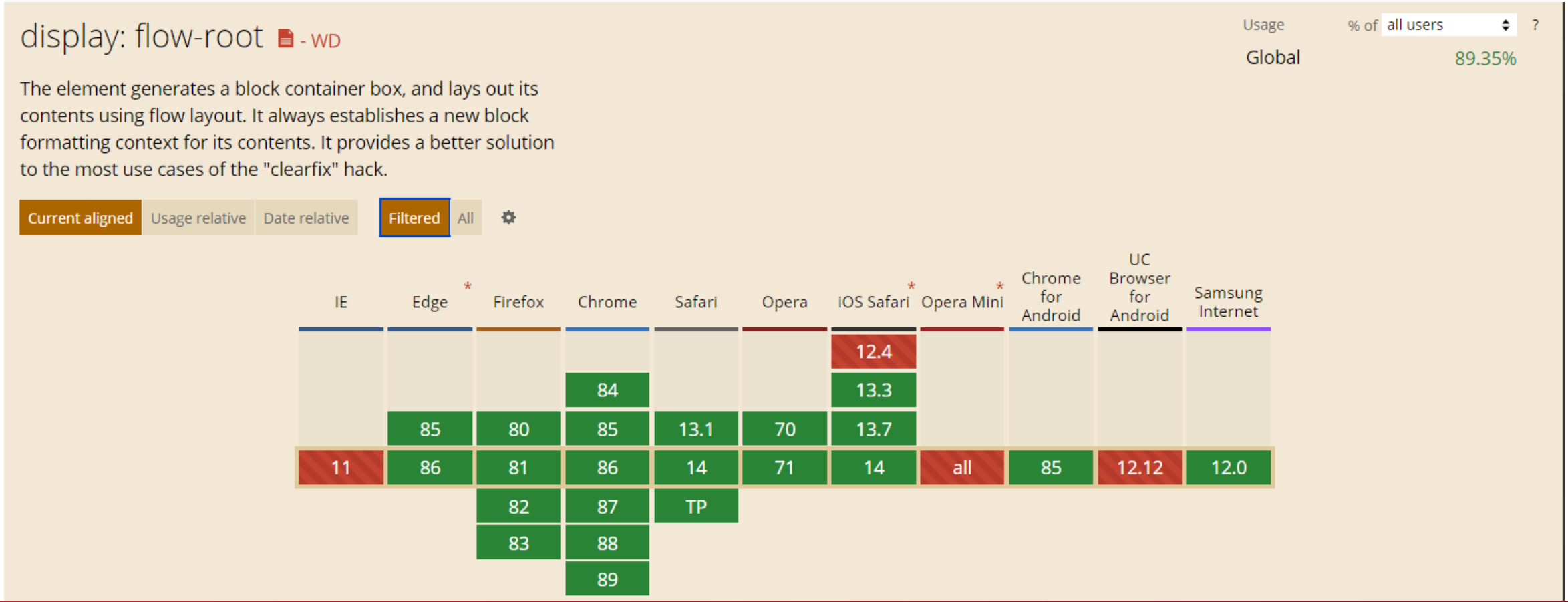


Mi definiál új block formatting contextet?

- Először is: a gyökérelem (`<html>`)
- A float-olt elemek (ahol a `float` nem **`none`**)
- Az abszolút vagy fixen pozícionált elemek
- Minden blokk elem, amin az `overflow` nem `visible`
  - pl. `hidden` vagy `scroll`
- A display egyes értékei:
  - `inline-block`, `table-cell`, `table-caption`, `[inline-]flex`
- Tehát a példában **a container nem volt önálló block formatting content**, de az overflow: hidden azt csinált belőle.

Nincs olyan, ami csak új block formatting contextet hoz létre?

- De van, a `display: flow-root`…
  - > de az elterjedtsége még nem az igazi.



Bootstrap

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Miért használjuk a Bootstrap-et?

- Könnyen tanulható és használható hála az igen részletes dokumentációnak
- Kiváló grid rendszer a reszponzív layout kialakításához
- Nem kell minden nulláról megírni
  - Alapvető HTML elemekre kész stílusok
  - Számos gyakran használt komponens design
  - Rengeteg időt lehet vele megtakarítani.
  - A 4-es verzió már flexbox alapú

Mit ad nekünk a Bootstrap?

- Először is formázást definiál jobbára szemantikus taghez, mint a `<blockquote>`, a `<mark>` vagy a headingek, és a `<body>` tipográfiájára is ad egy ízléses alapot.
- Számos shortcut osztályt tartalmaz (pl. `success`, `warning`, passzoló színekkel és ikonokkal)
  - Bár ezek egy része nem igazán „szemantikus”, pl. `text-center`
- Sok segítséget ad az űrlapok és táblázatok formázásához.
- Főleg kiindulási alapnak használjuk, leginkább a reszponzív CSS grid miatt, amit nyújt.

A Bootstrap grid

- **Mobile first:** mobilra mondjuk meg az elrendezést, majd ezt a képernyő növekedésével fokozatosan felülbírálhatjuk.
  - Alapvetően adaptív struktúrájú, de a töréspontok között bekapcsolhatunk folytonos méretezést
- A képernyőt sorokra és oszlopokra bontja.
  - Minden sor 12 azonos méretű oszlopból áll.
  - A sorok száma, mérete tetszőleges.
  - Egy konkrét tartalmi elem több oszlopon is átnyúlhat
    - de pl. "féloszlopon" alapvetően nem.
  - A sorok magasságát jellemzően nem "kézzel méretezzük", hanem a legmagasabb konkrét cellája határozza meg.

A grid használata

- A grid gyökere a `container` vagy `container-fluid` osztály
  - Ebbe `row` osztállyal dekorált konténereket teszünk…
  - …amikbe pedig kizárólag `col-*` osztályú elemek kerülnek.

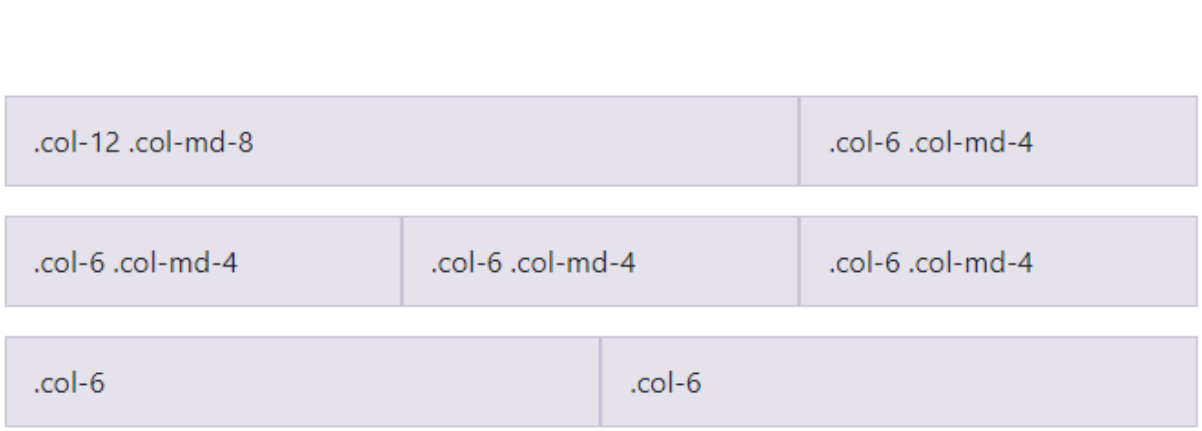
	xs <576px	sm ≥576px	md ≥768px	lg ≥992px	xl ≥1200px	xxl ≥1400px
Container `max-width`	None (auto)	540px	720px	960px	1140px	1320px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`	`.col-xxl-`
# of columns	12					
Gutter width	1.5rem (.75rem on left and right)					
Custom gutters	Yes					
Nestable	Yes					
Column ordering	Yes					

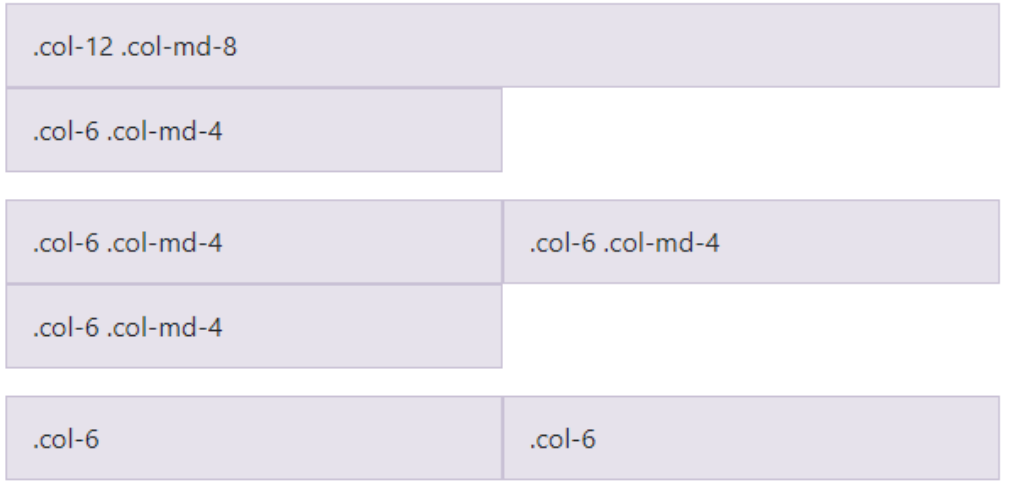
A col-* osztályok viselkedése

- Prefixszel megadjuk, melyik töréspont **fölött** alkalmazza
  - pl. a `col-sm-*` tableteken *és desktopokon is* érvényesül, de mobilon nem.
  - Ha nem adunk meg töréspontot, akkor minden felbontásnál az jut érvényre.
- A specifikusabb osztály felülírja a kevésbé specifikusat
  - pl. a `col-md-*` felülírja a `col-sm-*`-t, mert az kevesebb képernyőméretre érvényes.
- A * adja meg, hány oszlopot foglaljon el a cella
  - Az oszlopok száma *mindig* 12, mobilon is.
  - Az alapértelmezett cellaméret viszont 12.
    - Pl. ha egy cellára csak `col-md-*`-ot aggatunk, úgy kisebb képernyő mérten az alapértelmezett 12 lesz, tehát teljes szélességű lesz.

Mobil és desktop elrendezés

- A reszponzív oszlopok szélességekhez a különböző töréspontokra vonatkozó oszlop szélességeket együtt használni.
  - A `col-{méret}` minden felbontásra vonatkozik.
  - A `col-md-{méret}` viszont csak 768 px felett.
- Ugyanaz a layout 768px felett és alatt.





Automatikus oszlop szélességek

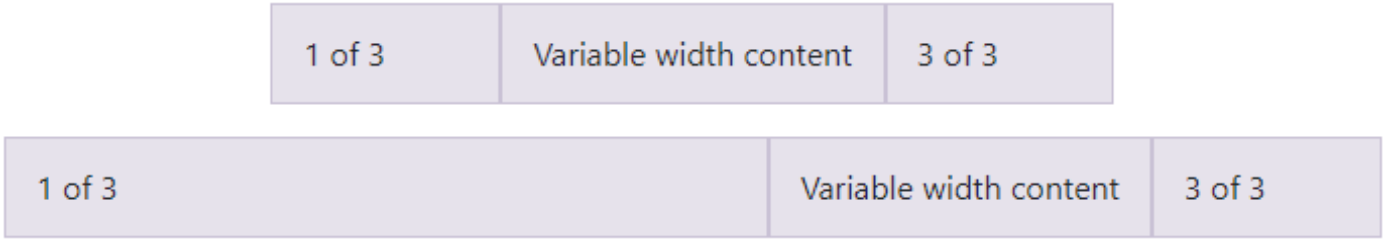
- A Bootstrap 4-ben bevezették a `col`-t is mint lehetséges oszlop szélességet.
  - Ha nem adjuk meg a töréspontot, és az oszlop szélességét, csak azt, hogy `col`, akkor az adott oszlop mérete automatikusan kerül meghatározásra.

1 of 2	2 of 2	
1 of 3	2 of 3	3 of 3

```
<div class="container">
  <div class="row">
    <div class="col">
      1 of 2
    </div>
    <div class="col">
      2 of 2
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col">
      2 of 3
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
</div>
```

Változó szélességű oszlop

- A `col-{md}-auto` bevezetésével lehetőség nyílik változó szélességű oszlopok definiálására is.
  - Ebben az esetben a tartalom mérete határozza meg az oszlop szélességét




```
<div class="container">
  <div class="row justify-content-md-center">
    <div class="col col-lg-2">
      1 of 3
    </div>
    <div class="col-md-auto">
      Variable width content
    </div>
    <div class="col col-lg-2">
      3 of 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-md-auto">
      Variable width content
    </div>
    <div class="col col-lg-2">
      3 of 3
    </div>
  </div>
</div>
```

Row columns

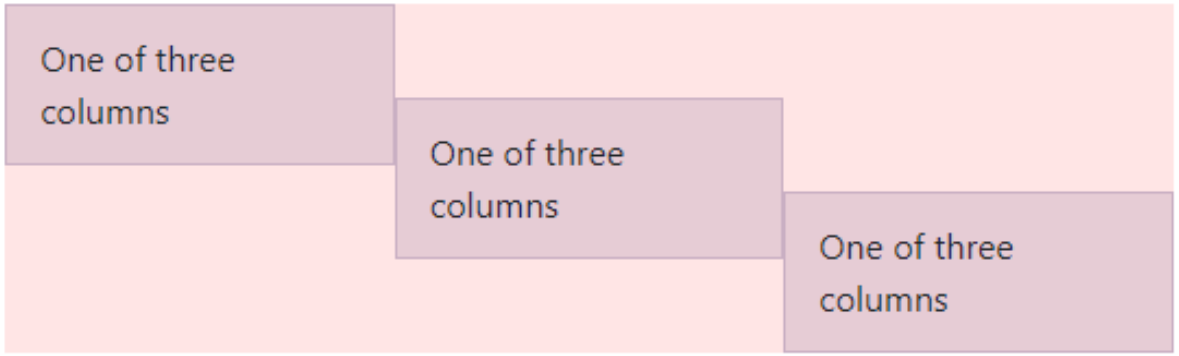
- A `row-cols-*` segítségével a sornál tudjuk megadni, hogy hány oszlopban jelenjen meg. Az egyes olszloppknál elég a `col`-t használni

```
<div class="container">
  <div class="row row-cols-2">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column	Column
Column	Column

Függőleges igazítás

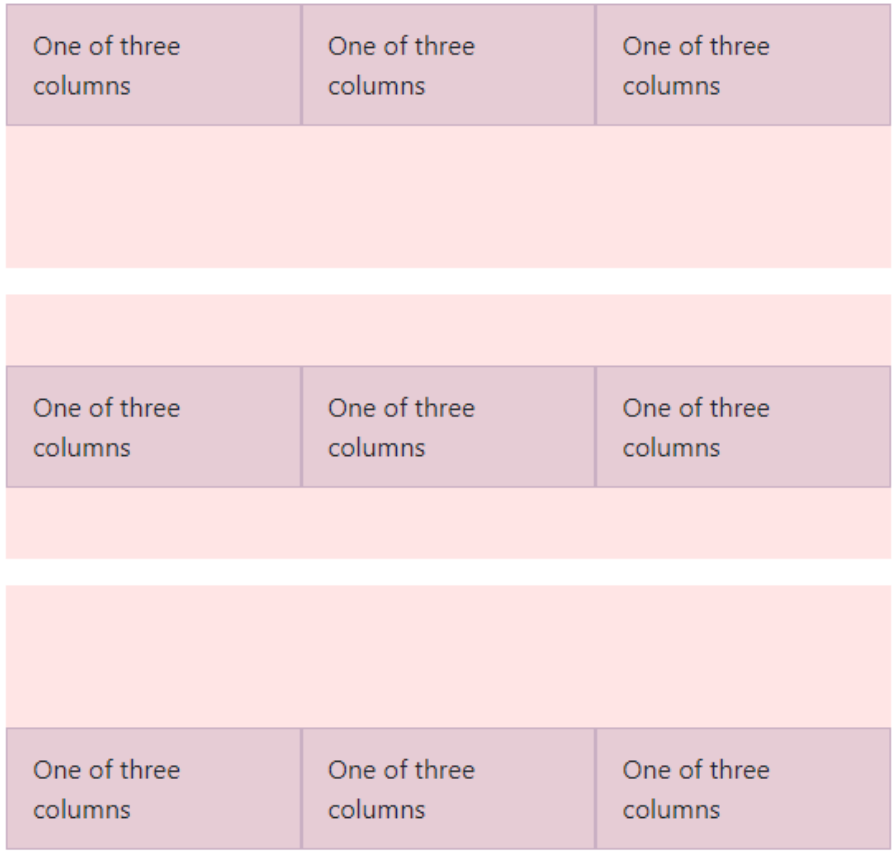
- A Bootstrap 4-től flexbox alapú ezért az oszlopokat függőlegesen is tudjuk igazítani.
- A flexbox a legfontosabb igazításhoz CSS osztályt is definiál
  - `align-self-start`
  - `align-self-center`




```
<div class="container">
  <div class="row">
    <div class="col align-self-start">
      One of three columns
    </div>
    <div class="col align-self-center">
      One of three columns
    </div>
    <div class="col align-self-end">
      One of three columns
    </div>
  </div>
</div>
```

Függőleges igazítás II.

- Nem csak az egyes cellákat, hanem a teljes sorokat is lehet függőlegesen igazítani.




```
<div class="container">
  <div class="row align-items-start">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
  <div class="row align-items-center">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
  <div class="row align-items-end">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
</div>
```

Row columns

- Oszlop kihagyása: offset-{md}-*
- Oszlop sorrend megadása: .order-{md}-*
- margin (.m-) / padding (.p-)
  - töréspont: sm / md / lg / xl / xxl
  - oldal: **t**op / **b**ottom / **s**tart / **e**nd / **x** / **y**
  - méretek: 0 – 5 vagy auto
- {tulajdonság}{oldal}-{töréspont}-{méret}
  - Pl: .me-md-3

HTML elemek reszponzív elrejtése / megjelenítése

- Ha egyes elemeket el szeretnénk rejteni adott képernyő méreten (pl: mobilon), akkor azt a display megadásával tudjuk megtenni.
  - `d-{value}`: mérettő függően mindig elrejti / megjeleníti.
    - Pl.: `d-none` vagy `d-block` vagy `d-flex`
  - `d-{breakpoint}-{value}`: adott képernyő méreten elrejti / megjeleníti.
    - Pl.: `d-sm-none` vagy `d-xl-flex`
  - A `{value}` a lehetséges display értékeket jelenti.
    - Pl.: none, inline, inline-block, block, flex…

Reszponzív képek

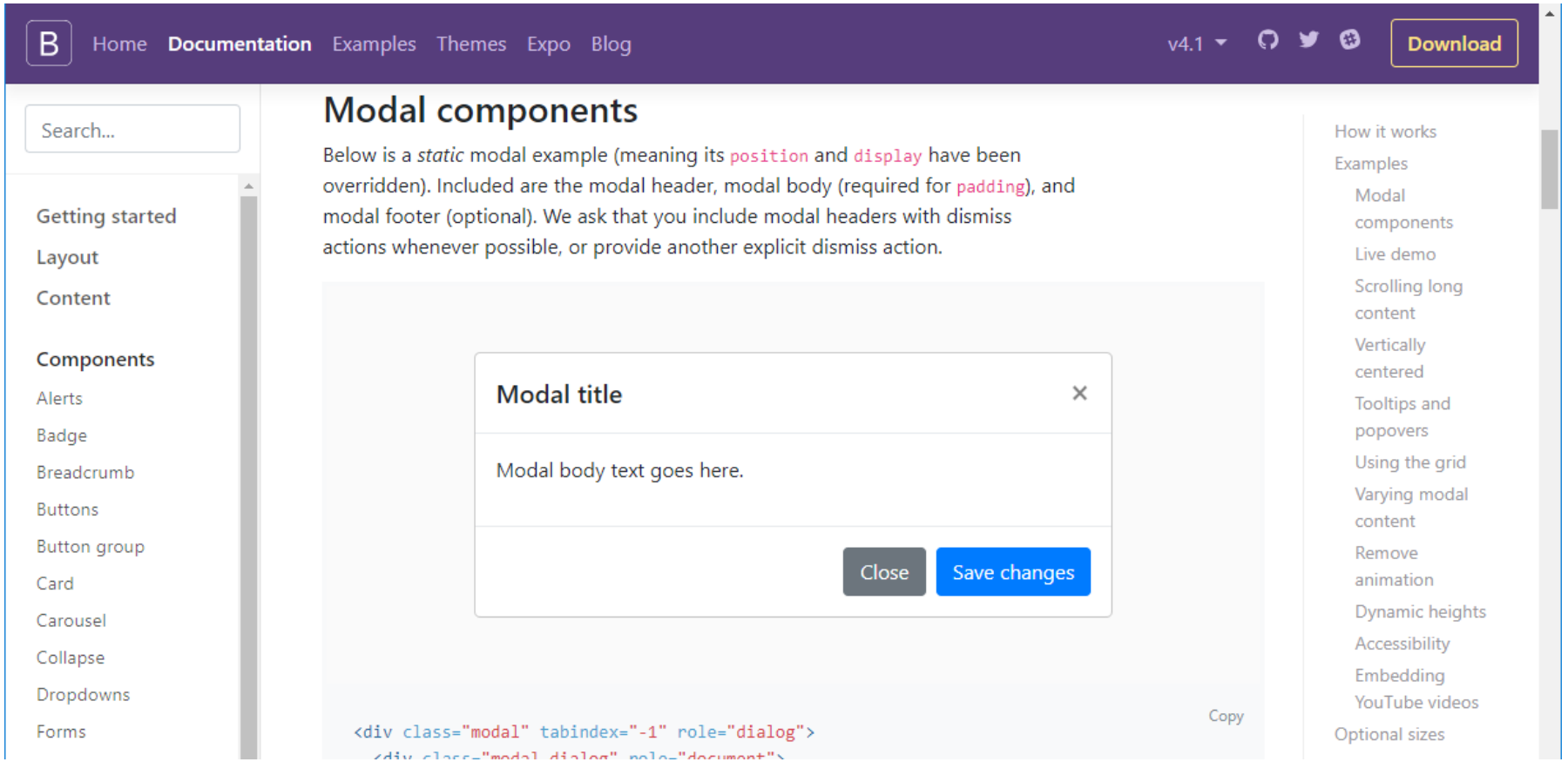
- **Cél:** a képek töltsék ki a szülőjük nyújtotta teret, de a képarány megtartása mellett.
- Megoldás (szélesség kitöltése):

```
max-width: 100%;
height: auto;
```

- Bootstrapben az `img-fluid` pont ezt csinálja.

Kész komponensek

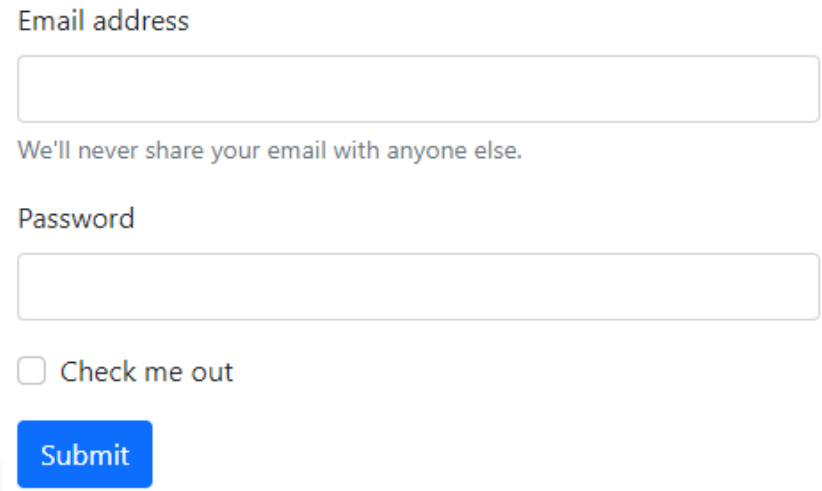
- Számos komponens design-t is kapunk készen.
  - Pl.: alert, badge, card, carousel, dropdown, modal…



Teljes lista: https://getbootstrap.com/docs/5.1/components/modal/

Űrlap kezelés

- A megfelelő HTML struktúra használatával űrlapokat is egyszerűen tudunk széppé varázsolni.




```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp">
    <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</div>
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

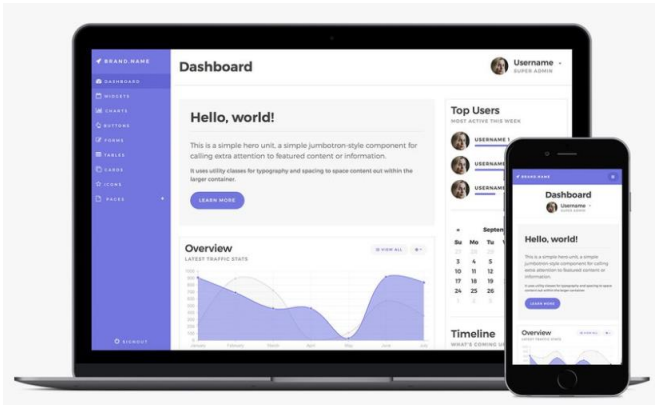

https://getbootstrap.com/docs/5.1/forms/overview/

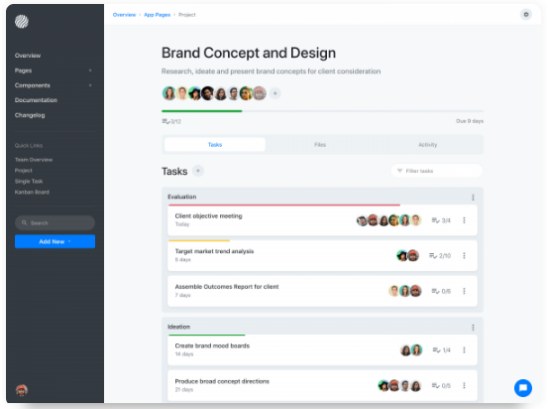
És milyen osztályok vannak még?

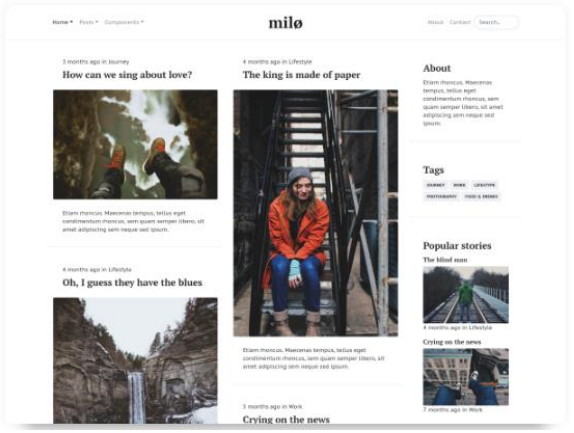
- Annyira sok formázást tartalmaz a bootstrap, hogy az összes osztályt nem lehet felsorolni.
- Azonban nagyon jó a dokumentáció, ahonnan bármit gyorsan ki tudunk keresni
  - https://getbootstrap.com/docs
  - Avagy érdemes lehet az alábbi cheat sheet-et használni.
  - https://bootstrap-cheatsheet.themeselection.com/
  - https://getbootstrap.com/docs/5.1/examples/cheatsheet/

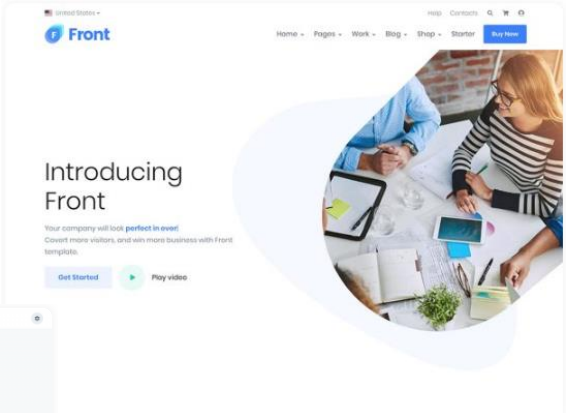
Bootstrap témák

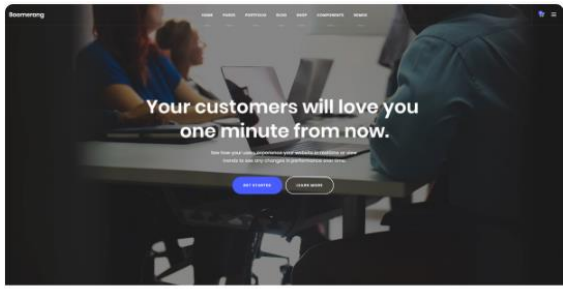
- Ha el szeretnénk térni az alapértelmezett megjelenéstől, akkor a témák használata egy jó választás lehet.

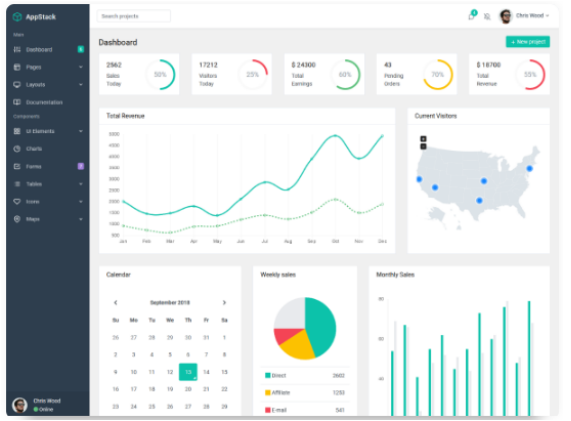












Flexbox

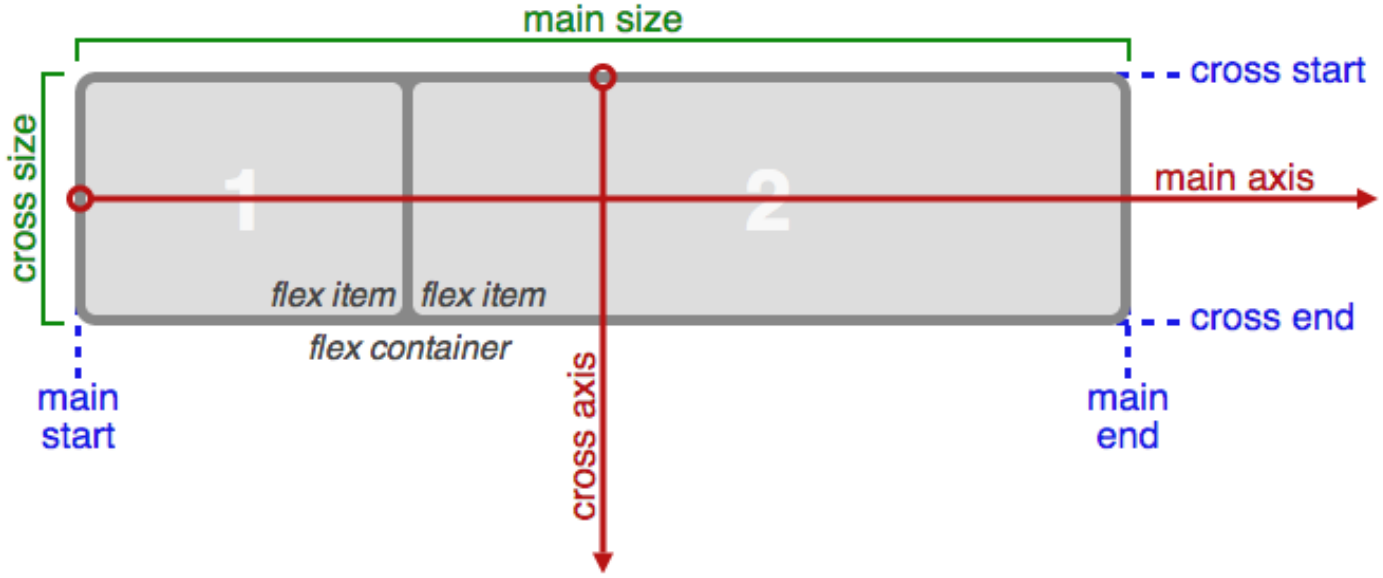
Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Mit old meg a flexbox?

- Teljes(ebb) kontrollt ad egy konténer gyerekei felett, reszponzív módon
  - Elemek igazítása egymáshoz és a konténer széleihez
  - Elemek automatikus méretezése arányosan
  - Elemek sorrendjének variálása



Flexbox tulajdonságok (konténer)

- `display: flex`
  - A konténeren kell megadni, hogy flexbox elrendezést használjon.
- `flex-direction`: gyerekek elrendezése
  - vízszintes (`row`)
  - függőleges (`column`) tengely mentén.
- `flex-wrap`: Sor-/oszloptörés engedélyezése.
- `justify-content`
  - Elemek rendezése, tagolása a főtengely mentén

justify-content

Flexbox: justify-content és flex-grow

Gincsai Gábor

Save Settings Change View

HTML

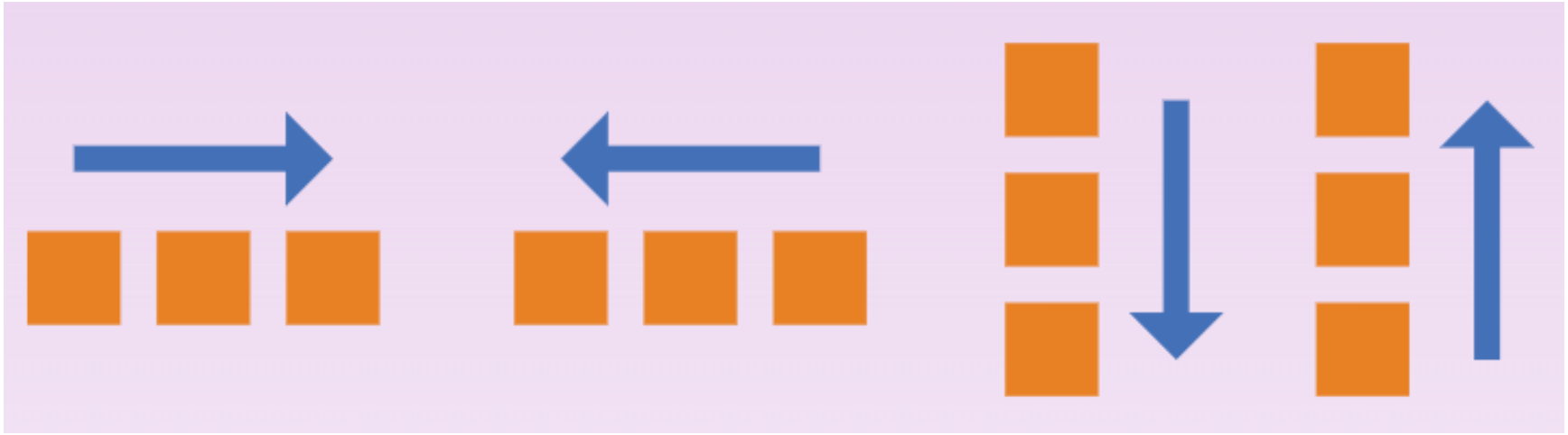
```
    <div style="background-color:coral;"></div>
    <div style="background-color:lightblue;"></div>
    <div style="background-color:khaki;"></div>
    <!-- Ő kapjon flex-grow: 1-et -->
    <div style="background-color:pink;" class="grow"></div>
</div>
```

CSS

```
#main {
    width: 400px;
    height: 70px;
    border: 1px solid #c3c3c3;
    display: flex;
    justify-content: space-between;
    /*justify-content: space-around;*/
    /*justify-content: space-evenly;*/
}
#main div {
    width: 60px;
    height: 60px;
}
/* .grow{ flex-grow: 1; } */
```

A flex-start és a flex-end értelmezése

	flex-start	flex-end
row	balra	jobbra
row-reverse	jobbra	balra
column	fentre	lentre
column-reverse	lentre	Fentre



align-items

- A gyerekek igazítása a főtengelyre merőlegesen.
  - Hasonlít a `justify-content`-re, de fontos különbség, hogy nem a *gyerekek között megmaradt* helyet osztja be, hanem a gyerekek és a *flexbox széle* közötti helyet.
  - Tehát ez akkor is működik, ha valamelyik gyerek „nyúlós” (mivel az a főtengely mentén nyúlik).
  - A gyerekek a kereszttengelyen is nyújthatók, erre van a `stretch` opció
    - Gyakori hiba azt hinni, hogy a `justify-content` rendelkezik `stretch` értékkel, de nincs neki, mivel a főtengelyi menti nyúlást a „gyerekek döntik el” (`flex-grow`)

Példa: align-items

Flexbox: align-items
Gincsai Gábor

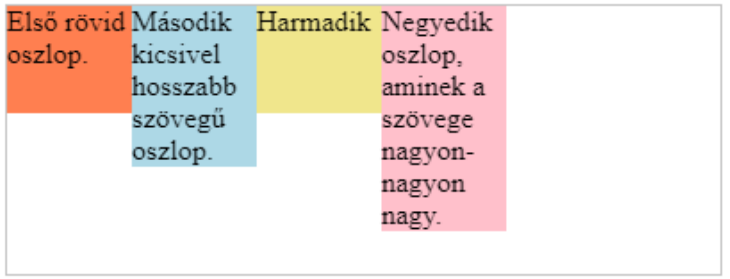
Save Settings Change View

HTML

```
<div id="main">
  <div style="background-color:coral;">Első rövid oszlop.</div>
  <div style="background-color:lightblue;">
    Második kicsivel hosszabb szövegű oszlop.
  </div>
  <div style="background-color:khaki;">Harmadik</div>
  <div style="background-color:pink;">
    Negyedik oszlop, aminek a szövege nagyon-nagyon nagy.
  </div>
</div>
```

CSS

```
#main {
    width: 400px;
    height: 150px;
    border: 1px solid #c3c3c3;
    display: flex;
    align-items: flex-start;
    /*align-items: center;*/
    /*align-items: flex-end;*/
    /*align-items: baseline;*/
    /*align-items: stretch;*/
}
```



align-content

- Többsoros flexbox *sorainak* igazítása a kereszttengely mentén.
  - Egysoros flexboxra nincs hatása.
  - Értékei kb. ugyanazok és kb. ugyanazt jelentik, mint a `justify-content` esetén, azonban itt van külön `stretch` érték is (ami a default), hiszen itt nem a főtengelyről, hanem a kereszttengelyről van szó, tehát nem a gyerekektől függ a „nyúlás”.

Flexbox gyerek tulajdonságai

- `flex-grow`: szabályozza, hogy a gyerek nyúljon-e – és a többiekhez képest milyen arányban –, ha marad hely a főtengely mentén. Alapértelmezett: 0 (nincs nyúlás)
- `flex-shrink`: összenyomható-e a gyerek, ha kevés a hely, és nem lehet sort törni; és ha igen, milyen arányban.
  - Alapértelmezett: 1
- `flex-basis`: a gyerek alap mérete nyújtás/összenyomás előtt, alapértelmezetten `auto`.
- `flex`: shorthand az előző háromra.
  - `flex: 1 1 auto` – teljesen „rugalmas”
  - `flex: 0 0 auto` – teljesen „rugalmatlan”

flex-basis részletek

- Alapértelmezett értéke az `auto`, ami a gyerek főtengely menti méretét (`width` vagy `height`) értékét veszi fel.
  - Ha az *is* `auto`, akkor a tartalomhoz igazodik a méret.
- Megadhatunk konkrét értéket is (pl. 25px, 33%), ezzel effektíve felülírjuk a főtengely menti méretet.
  - Ez akkor lehet hasznos, ha variáljuk a tengelyt (pl. mobil nézet), és emiatt simán a `width` vagy a `height` felülírása nem lenne elég rugalmas.

További flexbox gyerek tulajdonságok

- `align-self`: a gyerek „felülírhatja” saját magára a konténeren beállított `align-items` értéket.
- `order`: a gyerekek alapvetően abban a sorrendben jelennek meg, ahogy az a HTML-ben szerepel. Az `order`-rel ezen variálhatunk.
  - Negatív szám is lehet, alapértelmezetten 0.
- Van néhány „hagyományos” CSS tulajdonság, ami nem alkalmazható flexbox gyerekekre.
  - `float`, `clear`, `vertical-align`

Használhatom a flexboxot?

CSS Flexible Box Layout Module - CR

Method of positioning elements in horizontal or vertical stacks. Support includes all properties prefixed with `flex`, as well as `display: flex`, `display: inline-flex`, `align-content`, `align-items`, `align-self`, `justify-content` and `order`.

Current aligned | Usage relative | Date relative | Filtered | All

Usage	% of all users ?
Global	98.82% + 0.93% = 99.74%
unprefixed:	98.73% + 0.48% = 99.21%

Chrome	Edge *	Safari	Firefox	Opera	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	UC Browser for Android
						14.8			
						15.5			
105	105					15.6			
106	106	15.6	105			16.0			
107	107	16.1	106	91	107	16.1	18.0	all	13.4
108		16.2	107						
109		TP	108						
110									

Otthoni gyakorlás – Flexbox froggy

- Flexbox segítségével a békákat a saját színű tavi rózsájukra kell juttatni.
- A feladatokhoz megadják, hogy mely tulajdonságokat kell használni.