Mobil- és webes szoftverek

CSS

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

A CSS egyszerű nyelvnek tűnik

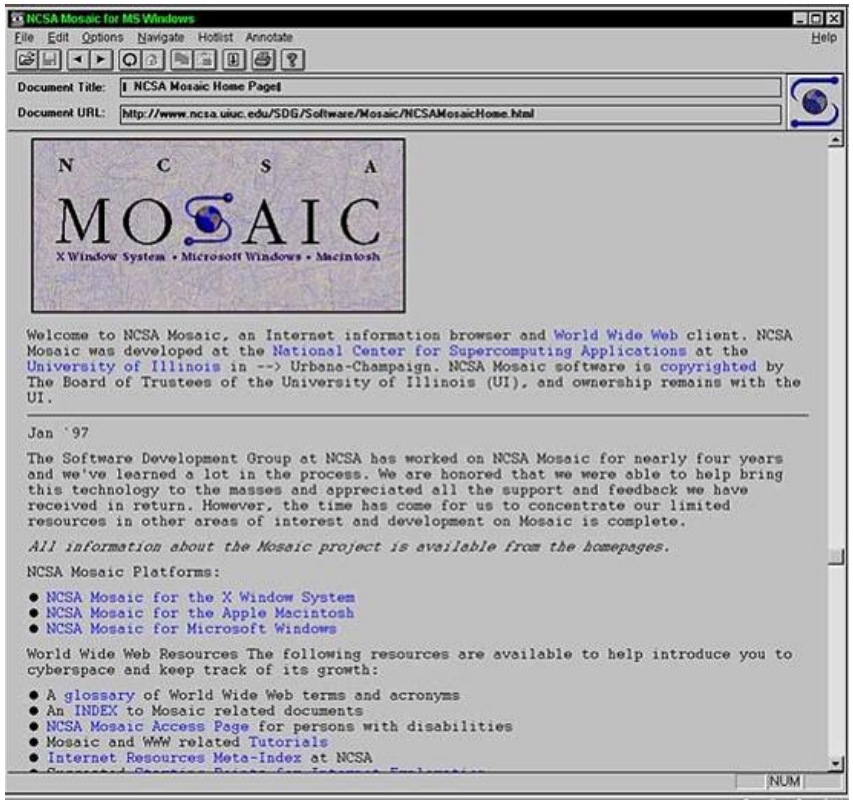
- Kulcs-érték párokat állítgatunk be fix értékkészletből, olyan magától értetődő tulajdonságokra mint szín vagy háttér.
- A szabályokat deklaratívan adhatjuk meg a dokumentum adott részeire vonatkozólag.

Mégis mi baj történhet?

- A tapasztalat az, hogy a kezdeti magabiztosságot rövidesen elharapódzó frusztráció követi.
- Amíg csak szöveget formázunk, minden rendben.
- Amint layoutot szeretnénk definiálni, amint pixelpontos dizájn kell, úgy érezzük, hogy hackelnünk kell.

Hogyan fajult ez idáig?

- **Ötlet:** Milyen jó lenne formázott dokumentumokat letölteni a hálózatról, amelyek akár **hivatkozásokat** is tartalmazhatnának egymásra.



Hogyan fajult ez idáig?

IBM Fri, 20 Dec 1996 01:09:44

[IBM News] [Products] [Industry solutions] [Services] [Support] [Research] [About IBM]

Headlines

- ['Tis the season...to do your holiday shopping on the Web]
- IBM announces completion of Edmark tender offer
- New DB2 Universal Database gives easy access to business information

Other voices: thinking beyond technology

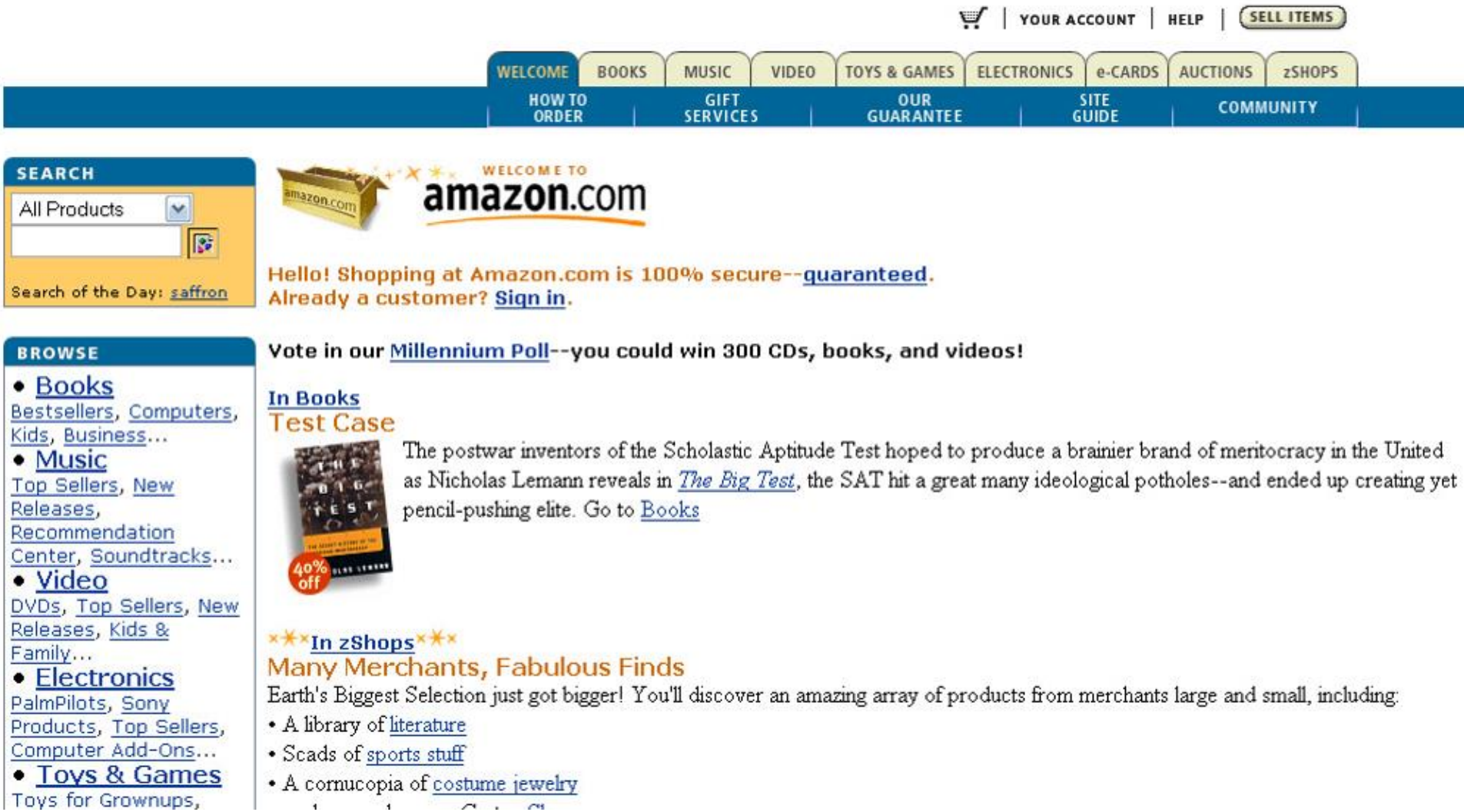
Computers & society: Electronic wizardry for Christmas
Business and the network: Europe Slowly Warming to The Internet

---

IBM planetwide

Take the Reins!
[IBM] | [Orders] | [Employment] | [Contact IBM] | [Legal]
Default, HTML 2.0, and HTML 3.2 versions of this document are also available.
This document generated at Fri, 20 Dec 1996 01:09:44 on sch-sp5.www.ibm.com

Hogyan fajult ez idáig?

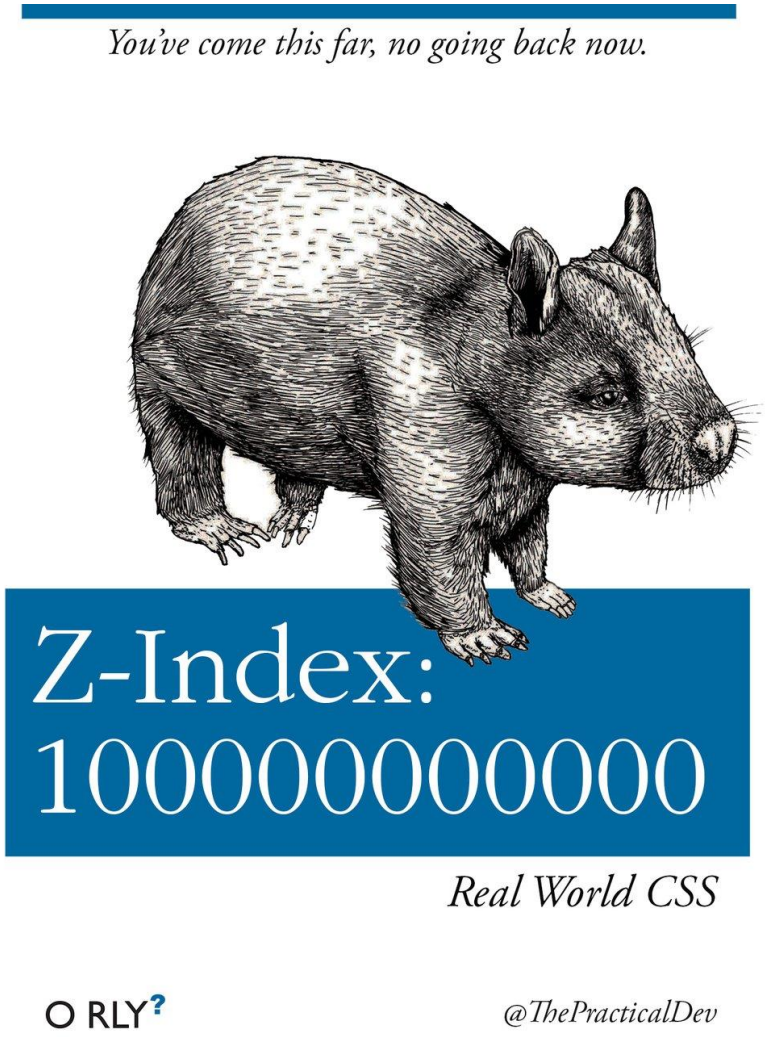


A weboldalak dokumentumok?

- A HTML 5 és a CSS 3 kezd elmozdulni ettől (kevesebbet is kell hackelni), de a HTML 4 + CSS 2 igenis egy **dokumentumformátum**.
- A jelszó: ismerd meg az ellenséget ☺
- Ahhoz, hogy megértsük, mi miért (nem) működik CSS-ben, meg kell értenünk, hogyan *szeretne* működni.
- Meg kell tanulnunk a weblapunkra dokumentumként tekinteni.

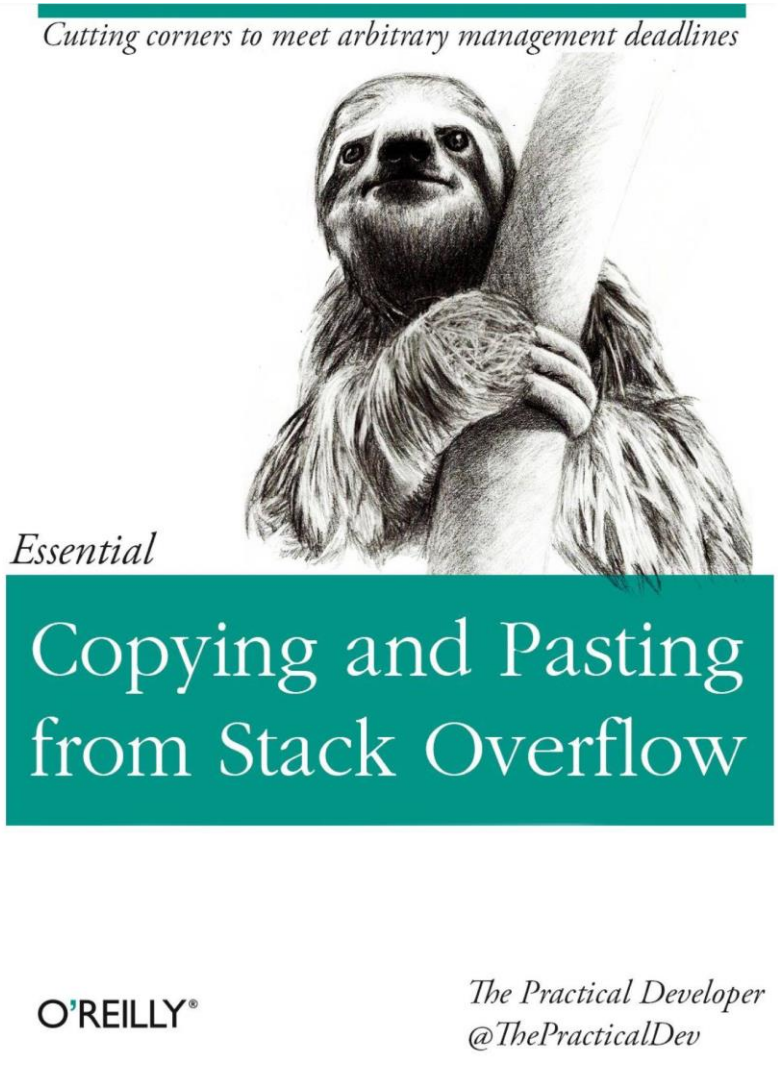
1. Megérteni a problémát

- „Ez miért nincs középen?”
  - Beállítottam a vertical-align: middle-t!
  - OK, tudom, hogy az szövegre vonatkozik, de még az se!
- „Ez miért nem 100% magas, ha egyszer beállítottam?”
- „Ez miért nem kerül a doboz fölé?”
  - Beírtam, hogy *z-index: 999999999;*



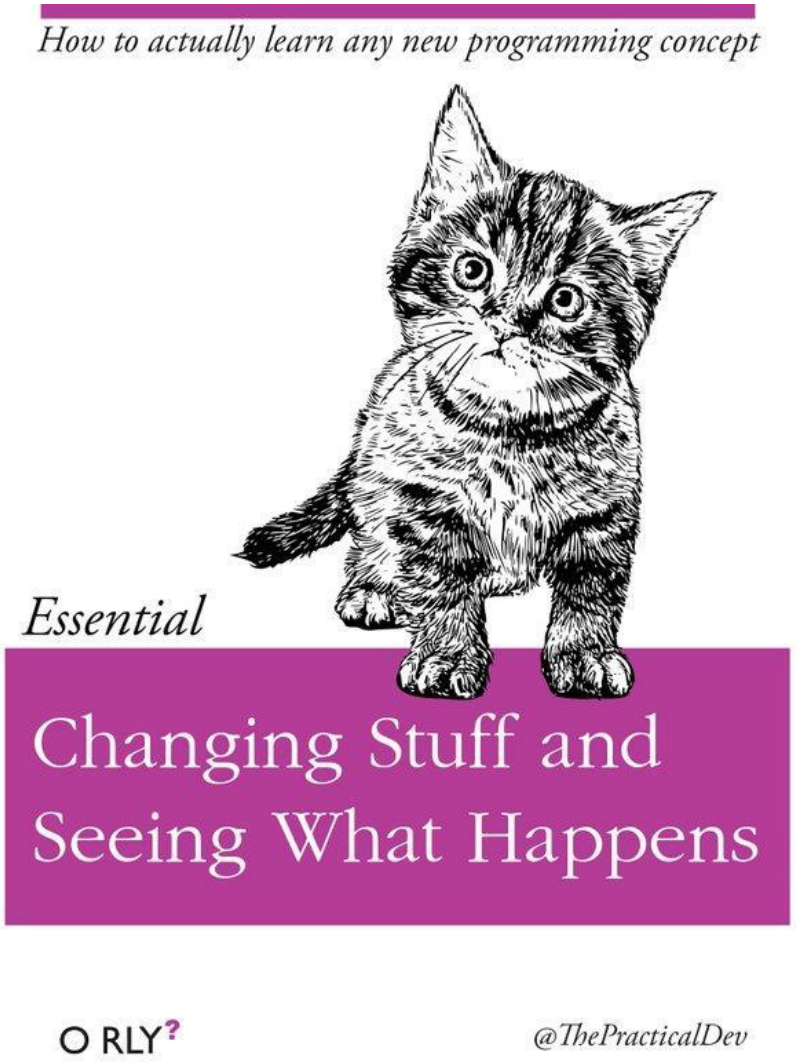
2. Megérteni a megoldást, mielőtt használjuk

- Ne fogadjunk el mindent ész nélkül, amit Stack Overflow-n találunk!

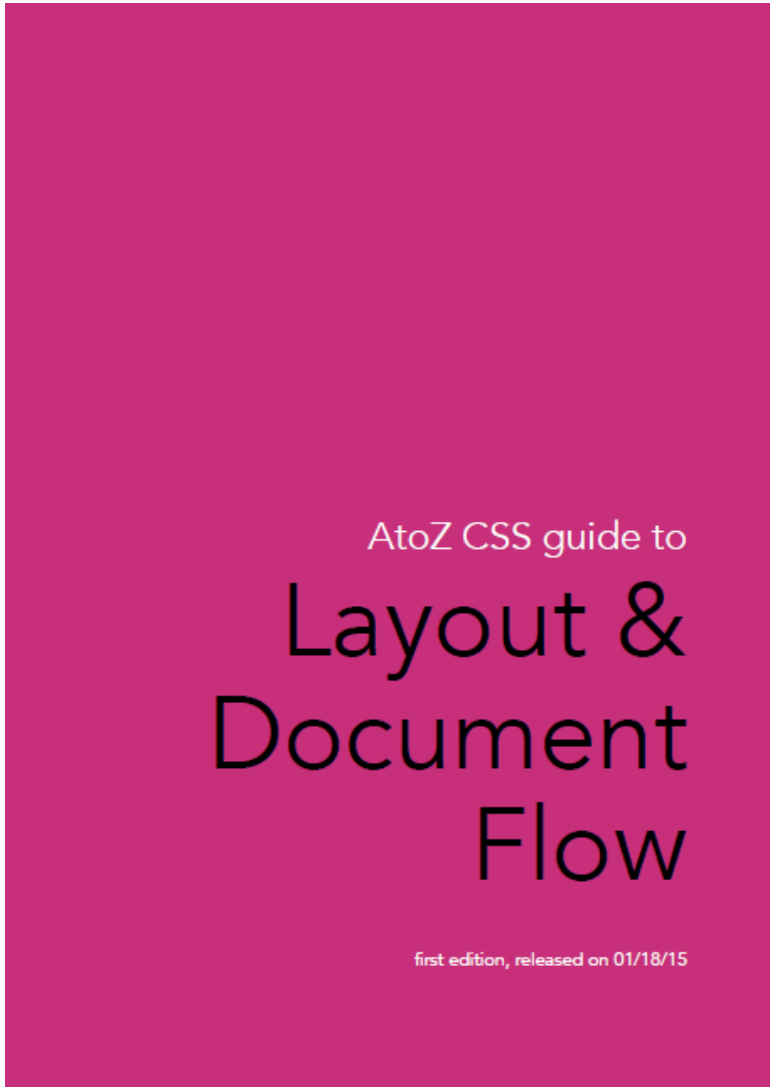


3. Megérteni a megoldást, mielőtt használjuk

- Ne „rúgdossuk addig, amíg jó nem lesz”!



Jót ~~lopni~~ tanulni nem szégyen



Mivel szembesülünk nap mint nap?

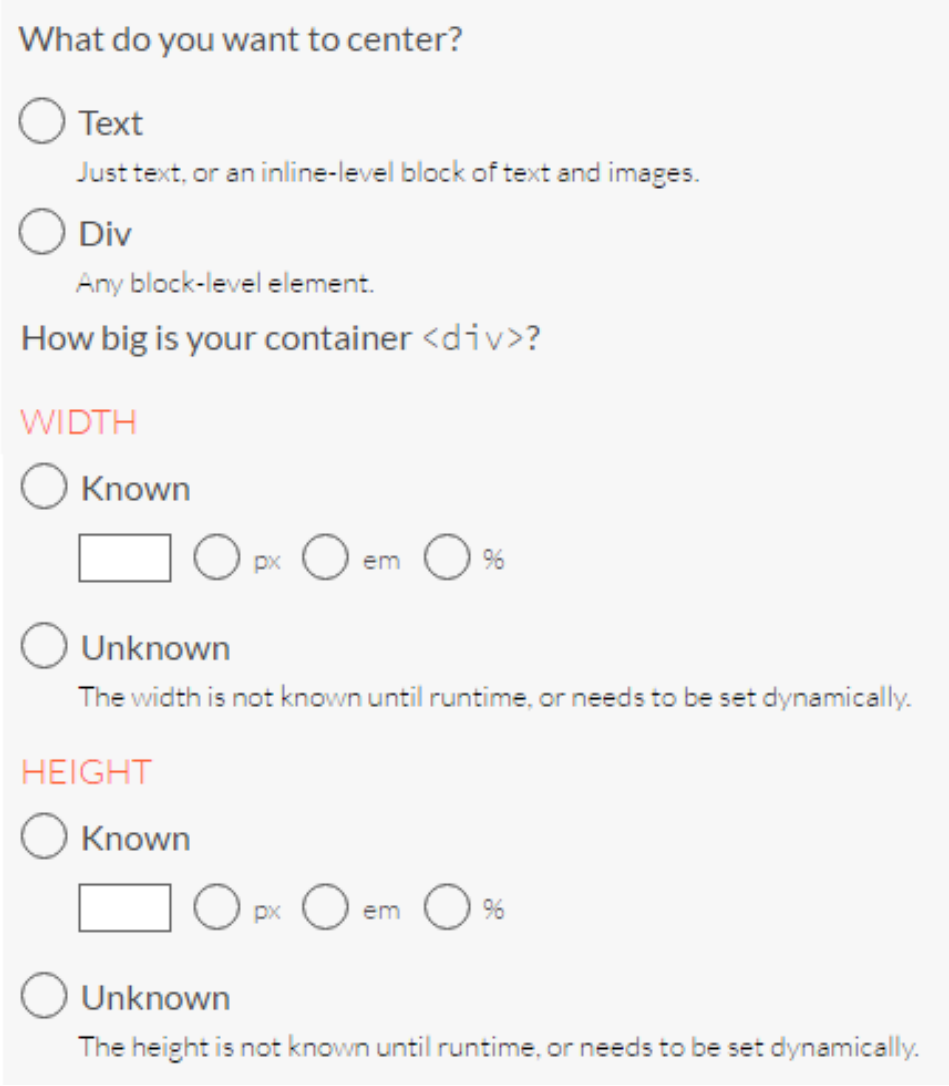
„Over the years, vertical centering has become the holy grail of CSS, as well as a popular inside joke between frontend professionals. The reason being that it has all of the following properties at the same time:

- *It's very frequently needed.*
- *It sounds exceedingly easy and simple in theory.*
- *It used to be incredibly difficult in practice, especially for elements of variable dimensions."*

(Lea Verou - CSS Secrets)

Persze, hogy van megoldás…

- De nincs egy helyes út…
- CSS-ben bármi a probléma, a válasz: **„attól függ…”**



Cascading Style Sheets

Alapok

CSS - Cascading Style Sheets

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

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>
	…
	

Saját osztályokra, amire a class attribútummal hivatkozunk	

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

Tetszőleges azonosítóra, amire az id attribútummal hivatkozunk	

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

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

Összetett selectorok

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

nav > .main {
  color: red;
}
	<nav>
	  …
	</nav>

Tagen belüli osztály (leszármazottak)	

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

Tag és osztály szerepel egyszerre	

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

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

További selectorok

- Univerzális selector:

```
*{ color: red;}
```

- Attribútum:

```
a[title] { color: red; }
```

- Testvér:

```
img ~ p { color: red; }
```

- Közvetlen utána következő testvér:

```
img + p { font-style: bold; }
```


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

Pszeudo osztályok

- 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

- Új elem létrehozása első gyerekként

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

- Új elem létrehozása utolsó gyerekként

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

- Kiválasztott elemek (pl.: szövegrész)

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

- Első betűre egyedi megjelenés

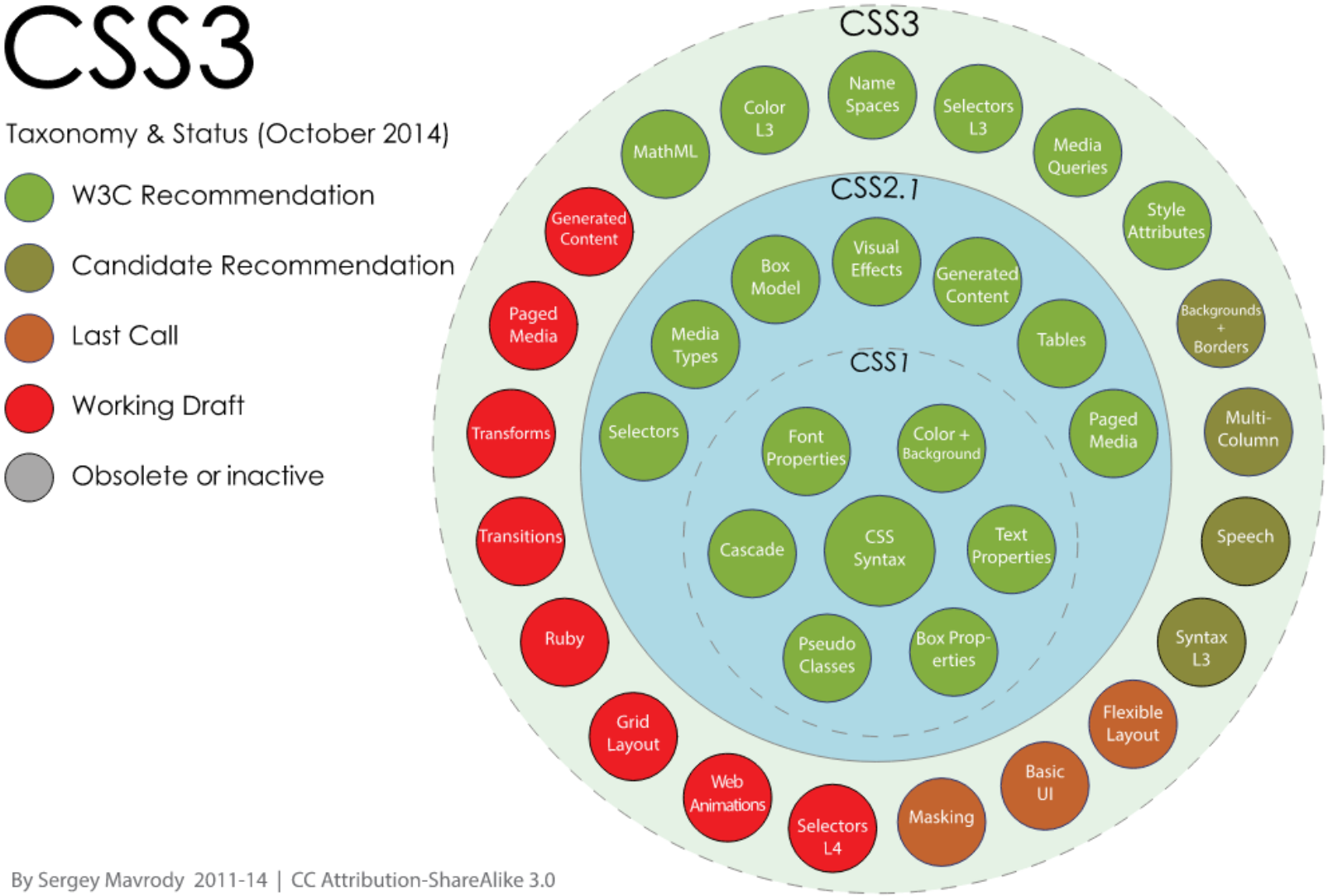
```
::first-letter{ font-size: 40px; }
```


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

CSS3

Böngésző támogatottság:

CSS3 támogatottság...



Támogatottság ellenőrzése

- Fejlesztés közben nézünk utána
  - futásidőben "pótolni" a CSS hiányokat többnyire jóval nehezebb, mint a HTML5-ös JS API-kat.
    - Ha nincs mondjuk LocalStorage, attól az oldal alapvetően még működőképes maradhat.
    - 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?

- JS-ből próbáljuk utánozni a működést ("polyfill")
- Egyszerűbb CSS-sel helyettesítjük
  - gradiens helyett kép
  - *A*z "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
  - még csak kevés böngésző támogatja
  - "trükkösen" kell használni.

A @supports blokk

- Ha egy `propery: érték` pár támogatott, érvényre jutnak a szabályok, ha nem, nem.
- Ha egy böngésző *egyáltalán nem* ismeri a `@supports` blokkot, a feltételtől függetlenül figyelmen kívül hagyja a tartalmát

```
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, ahogy már beszéltünk róla, 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: gradiens

```
body {
  /*Safari 5.1-6*/
  background: -webkit-linear-gradient(
    left, rgba(255,0,0,0), rgba(255,0,0,1));
  /*Opera 11.1-12*/
  background: -o-linear-gradient(
    right, rgba(255,0,0,0), rgba(255,0,0,1));
  /*Firefox 3.6-15*/
  background: -moz-linear-gradient(
    right, rgba(255,0,0,0), rgba(255,0,0,1));
  /*Standard*/
  background: linear-gradient(
    to right, rgba(255,0,0,0), rgba(255,0,0,1));
}
```

Másik példa: flexbox

```
body {
  display: -webkit-box;
  display: -moz-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  display: flex; /* Ez bizony... */
  -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; /* ...3 property... */
  -webkit-box-align: stretch;
  -moz-box-align: stretch;
  -webkit-align-items: stretch;
  -ms-flex-align: stretch;
  align-items: stretch; /* ...de 17 sor kód! */
}
```

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…

Kaptam egy dizájnt, hogy álljak neki a HTML-nek?

1. fázis: alapvető hierarchia

Az alapvető tartalmi elemek meghatározása

- Mi a legfontosabb tartalmi elem?
- Mi legyen a legfelső szintű címsor?
- Miket tekinthetünk alsóbb szintű címsoroknak? Hol lehet rájuk szükség?
- A tartalom mely részét kezeljük listaelemekként?
- Hol lehetnek kereszthivatkozások képek, szövegek és blokkok között?

2. fázis: csoportosítás

Az elemek közti összefüggések meghatározása

- Mi tartozik össze mivel?
- A megfelelő strukturális elemek megválasztása, és általuk az alapelemek csoportosítása
- Itt már elkezdhetünk gondolkodni azon, mik kerülnek majd külön "oszlopba" (egymás mellé), és ez hogyan rendeződik majd át keskenyebb kijelzőkön

3. fázis: készüljünk fel a változásra…

- Most, hogy a HTML tartalomleíró szerepét kipipáltuk, jöhet a layout meghatározó szerepe.
- Amennyire lehet, ezt CSS szabályok megfogalmazásával (és vele szinkronban class attribútumok segítségével) próbáljuk megoldani.
  - …de a HTML többi része sincs kőbe vésve – csak kiindulási alap, amit eddig csináltunk.
  - Nem baj (és szinte elkerülhetetlen) néhány extra -et vagy -t beszúrni, ahova kell.
  - A lényeg, hogy ne ok nélkül tegyük – ezért nem indítunk a layout aspektussal.

Vizuális építő elemek

A befoglaló négyszög / doboz

- Az oldalon minden elem egy doboz.
  - Ha kör vagy háromszög alakú is, vagy egy átlátszó hátterű kép, akkor is egy befoglaló négyszögön belül (bounding box) helyezkedik el.
- A befoglaló négyszög határozza meg, mekkora helyet foglal el egy elem az oldalon
- A befoglaló négyszög alapvetően láthatatlan, de egy kis egyedi CSS-sel láthatóvá tehetjük.

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

*Az !important azért kell, mert amúgy az univerzális * selector elég gyenge.*

A befoglaló négyszög / doboz

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.
  - Következésképp 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 blockok egymásba ágyazhatóak, 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 jobbr~~ a dokumentum szövegirányát követi. (i18n ftw)

Az egyes típusok által felvehető tulajdonságok

- Az imént leírt 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
  - `text-align` tulajdonságra (vízszintes rendezés)
  - 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 átállításával
  - Erre még visszatérünk
- Egyes CSS propertyk "kiszakítják" az inline elemeket a dokumentumfolyamból, *effektíve* blokkszerűvé alakítva őket
  - pl. `float`, `position`…
- Így a `width`, `height` stb. beállíthatóvá válik rajtuk.

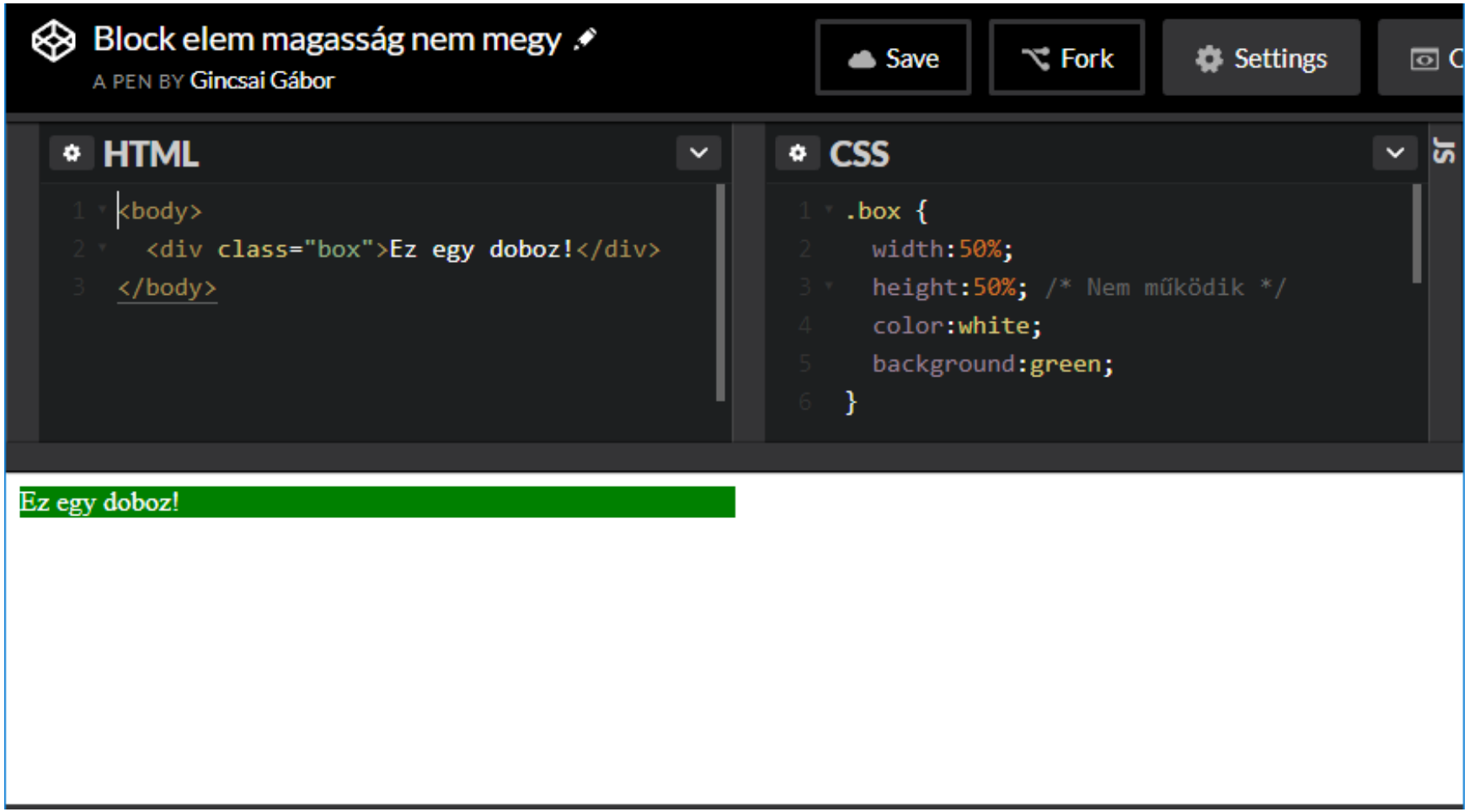
Blokk elemek – width

- 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 `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**

Block elem magassága



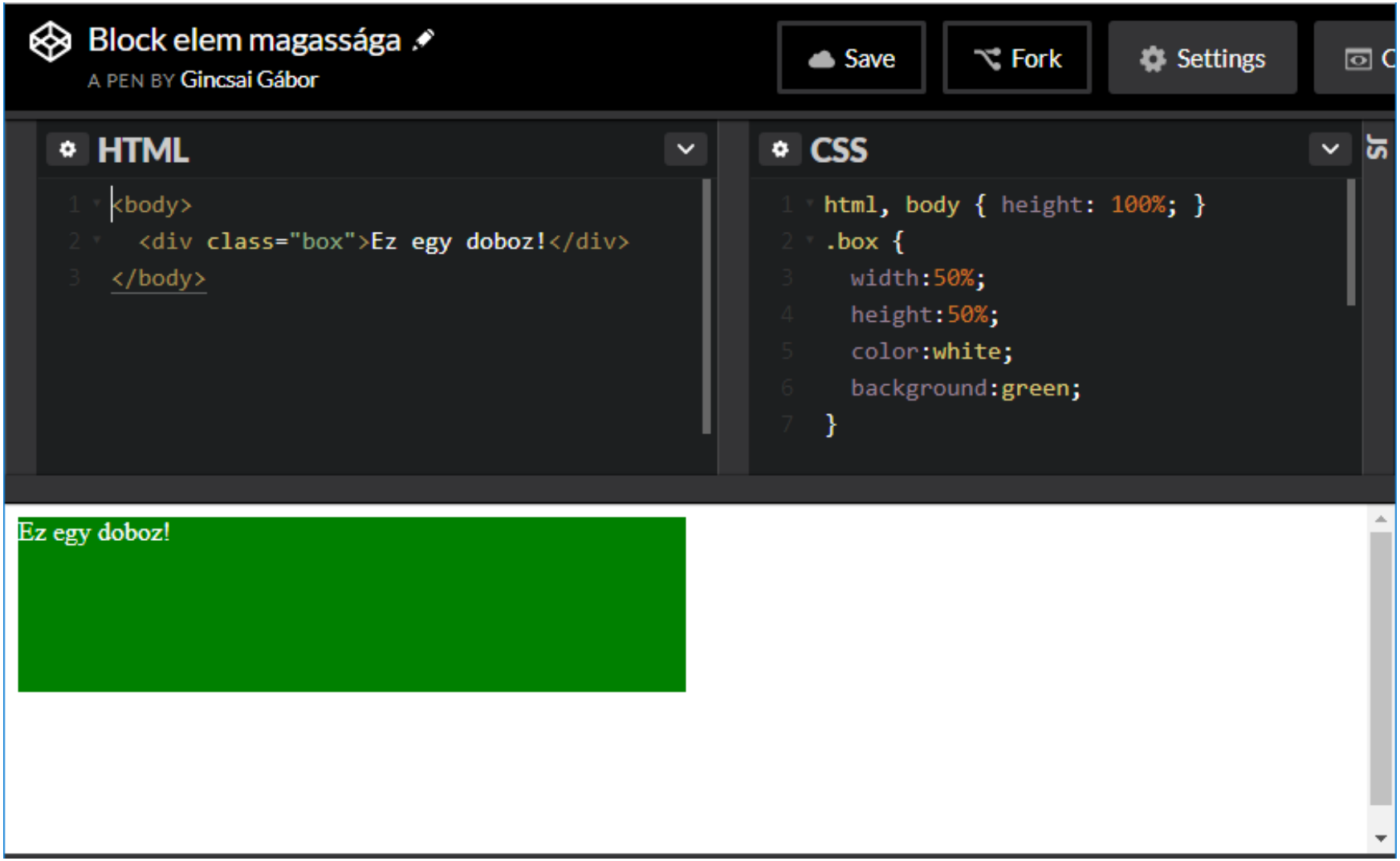
https://codepen.io/gincsai/pen/rGqbJQ

Miért nem működik?

- Mit adtunk meg?
  - Beállítottuk a dobozon a 80% 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



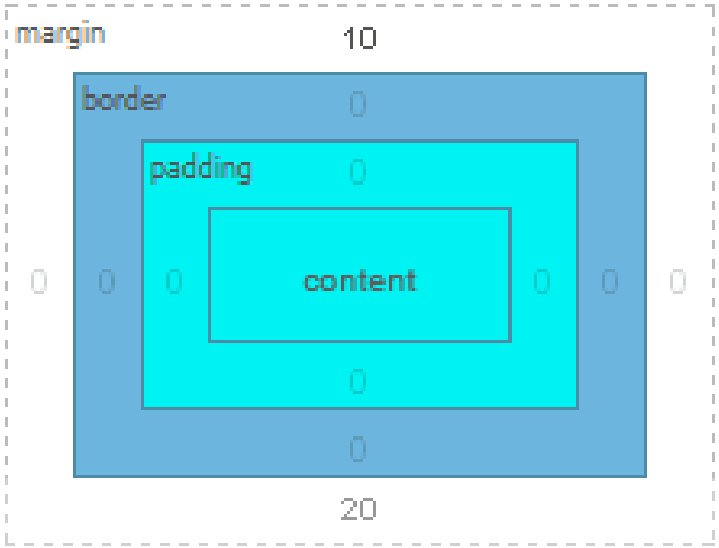
https://codepen.io/gincsai/pen/PJxBGx

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 fölöttünk 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

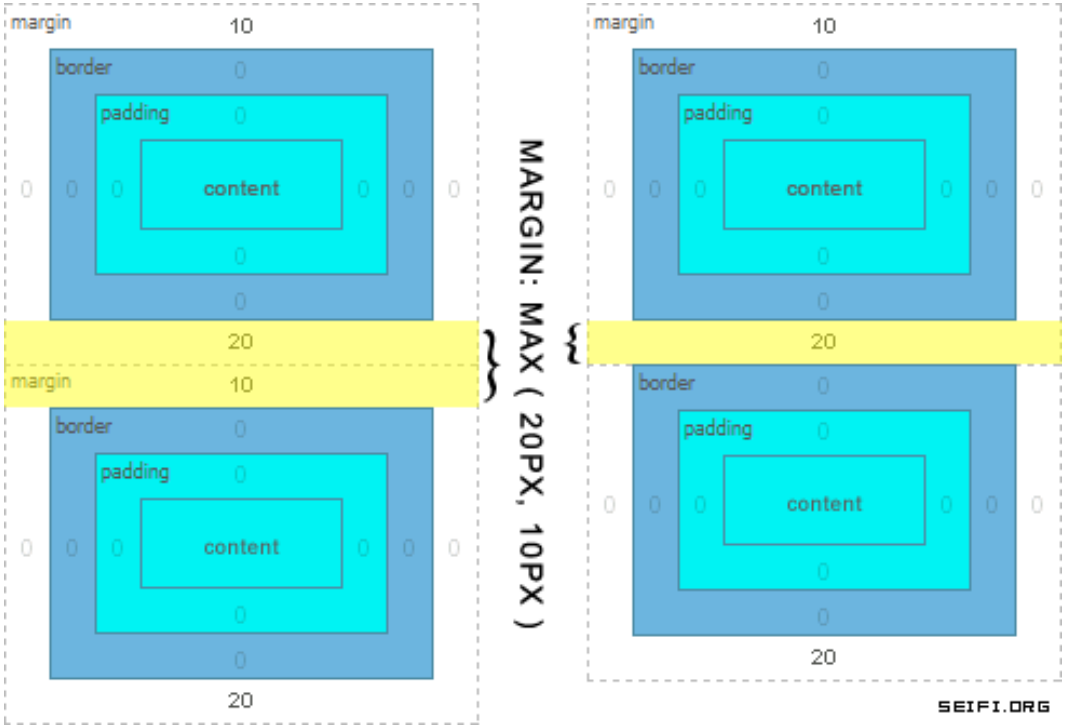
Blokk elemek – margin és padding

- 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

Margin collapse – hogyan szabaduljunk meg tőle?

- Akadályozzuk meg, hogy a margók összeérjenek
  - Vezessünk be közéjük bordert.
  - Vezessünk be közéjük paddingot.
  - Vezessünk be fura háziszabályokat pl.:
    - margót mindig csak lefele,
    - paddingot mindig csak fölfele definiálunk
- Kényszerítsük ki új blokk formázási kontextus létrejöttét
  - Erre még visszatérünk :)

Margin collapse

Margin collapse

A PEN BY Gincsai Gábor

Save | Fork | Settings | Change V

HTML

```
<body>
  <div class="box">
    <p>Ez egy paragrafus</p>
  </div>
</body>
```

CSS

```
body {
  border:1px solid black;
}
.box {
  margin-top:20px; /* összevonódik a gyerekével! */
  /* border:1px solid red; */ /* van border, akkor nem! */
}
.box p {
  margin-top:20px;
  margin-bottom:20px;
  background:#cc3f85;
}
```

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

Mi baj lehet a paddingból?

04 - Mi baj lehet a paddingból

A PEN BY Gincsai Gábor

Save | Fork | Settings | Change

HTML

```
<body>
  <div class="left">Bal</div>
  <div class=right>Jobb</div>
</body>
```

CSS

```
div{
  float: left;
  width: 50%;
  height: 100px;
  /*padding: 2px;*/
  /*box-sizing: border-box;*/
  /*width: calc( 50% - 4px );*/
}
.left{
  background-color: green;
}
.right{
  background-color: yellow;
}
```

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

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/

Mi volt a baj?

- 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 lehet megoldani!
  - Csak úgy, ha 50%-ról fix (px) szélességre váltunk, amiből ki tudjuk vonni a dobozonként 4 pixelnyi paddingot!
- CSS3-ban szerencsére van megoldás
  - **width: calc(50% - 2px)**

Inline vagy block elem?

- Az inline-ban az a jó, hogy
  - az elemek szépen egymás mellé kerülnek,
  - és reagálnak a vertical-align propertyre.
- A block-ban az a jó, hogy
  - játszhatunk a margókkal, paddinggal, méretekkel.
- Nem lehetne mindkettőből kérni egy kicsit?

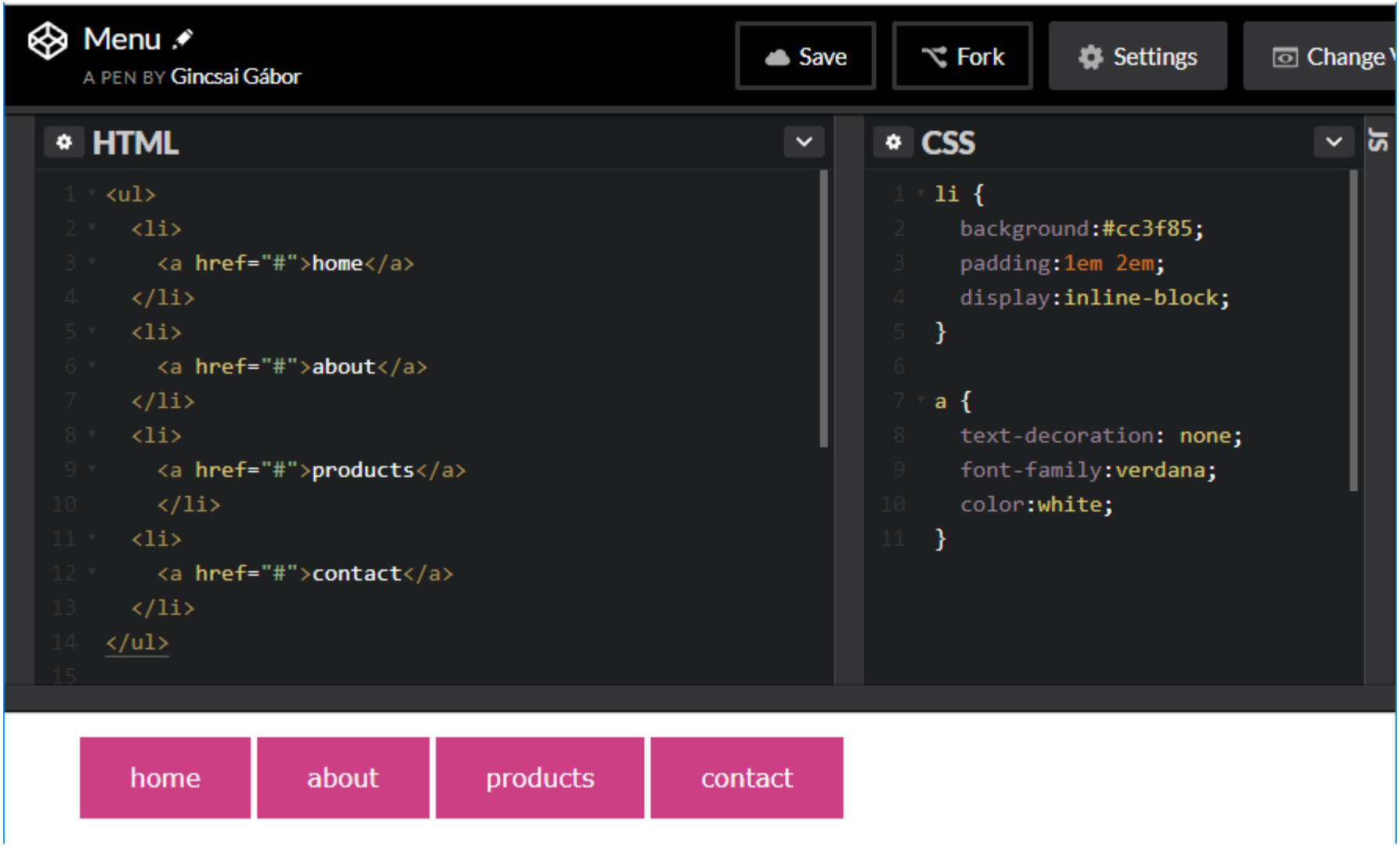
display: inline-block

- Az elem az inline flow része marad.
- De emellett blokk tulajdonságok is beállíthatóak rajta!
- Kiválóan alkalmas pl. egy menüben az elemek egymás mellé helyezésére.
- Nem „hülyíti meg” a környező elemeket, mint a `float`.
- A `vertical-align` is beállítható rajta.

Ilyen szép menünk lesz!


```
<ul>
      <li>
            <a href="#">home</a>
      </li>
      <li>
            <a href="#">about</a>
      </li>
</ul>
<main>
      <img src="…" />
</main>
```

Vagy mégsem?

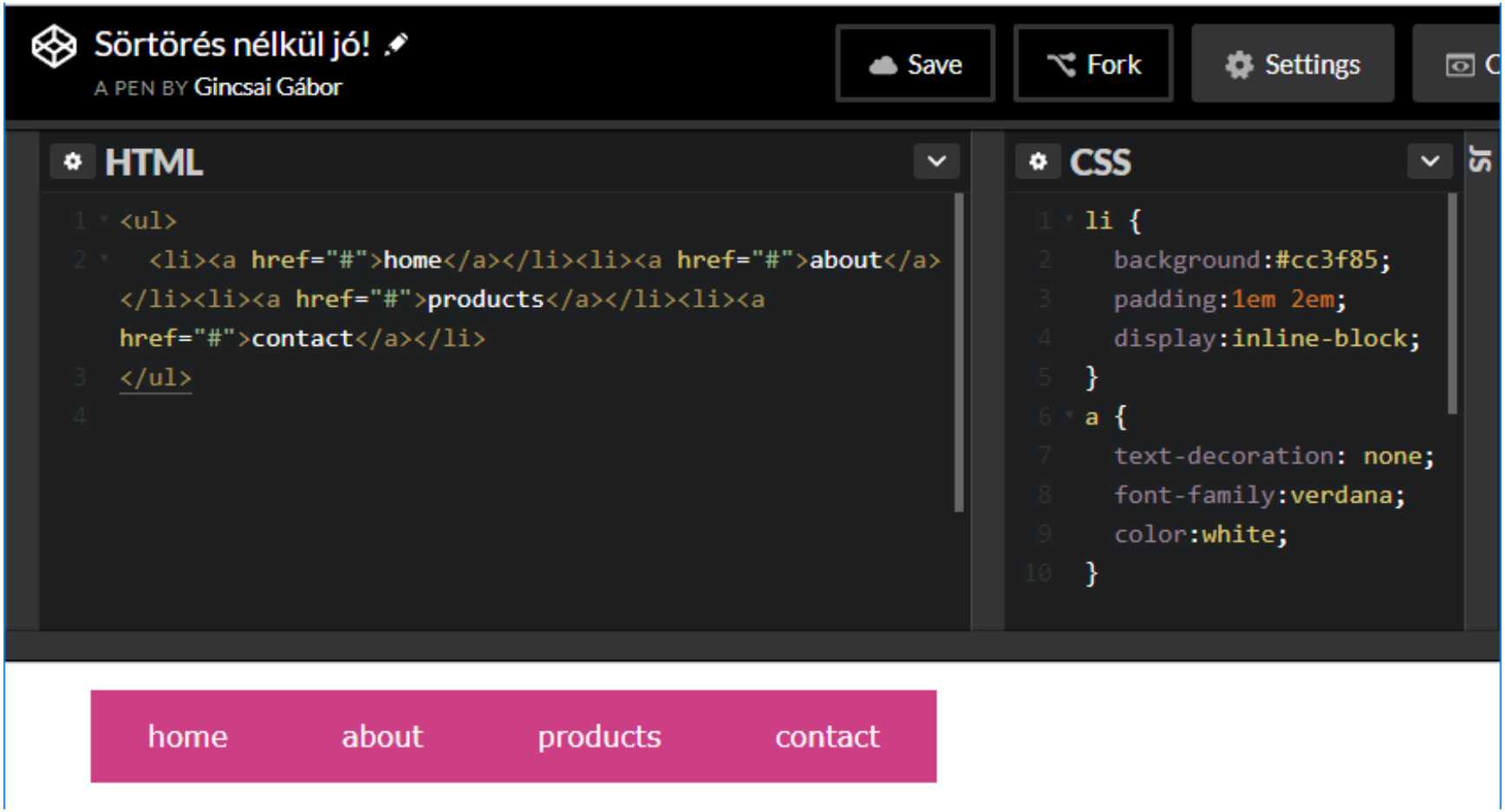


https://codepen.io/gincsai/pen/ZXmmbb

Mi történt?

- Az inline flow nem dobja el a whitespace karaktereket, csak ***egyetlen*** *szóközzé* tömöríti őket.
- Mivel a listaelemeink az inline flow részei, a köztük szereplő szóköz egy karakter széles térközként jelenik meg
- **Dehát hol van itt szóköz, kérem?!**

Megoldás 1: Nem kell sortörés



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

Megoldás 2: Kommentek

Kommentezzük ki a sortörést!

A PEN BY Gincsai Gábor

Save | Fork | Settings

HTML

```
<ul>
  <li><a href="#">home</a></li><!--
  --><li><a href="#">about</a></li><!--
  --><li><a href="#">products</a></li><!--
  --><li><a href="#">contact</a></li>
</ul>
```

CSS

```
li {
  background:#cc3f85;
  padding:1em 2em;
  display:inline-block;
}
a {
  text-decoration: none;
  font-family:verdana;
  color:white;
}
```

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

HTML hack nélkül nem megy?

- Dehogynem, elég a CSS-t hackelni!
- Mit tudunk?
  - A térközt a listaelemek közti szóköz okozza.
  - A térköz mérete a listaelemek közti szóköz betűmérete.
- Következtetés
  - Ha 0 lenne a szóköz betűmérete, nem lenne térköz.
  - A listalelemek közti szöveg betűméretét kell átállítanunk!
- Megoldás
  - A `<ul>` betűméretét le kell nullázni
  - Mivel ezt a <li> is megörökölni, ott helyre kell állítani!

A megoldás CSS-sel

CSS hack

A PEN BY Gincsai Gábor

Save | Fork | Settings

HTML

```
<ul>
  <li><a href="#">home</a></li>
  <li><a href="#">about</a></li>
  <li><a href="#">products</a></li>
  <li><a href="#">contact</a></li>
</ul>
```

CSS

```
ul{
  font-size: 0
}
li {
  font-size: 1rem;
  background:#cc3f85;
  padding:1em 2em;
  display:inline-block;
}
a {
  text-decoration: none;
  font-family:verdana;
  color:white;
}
```

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

Mi inline-block alapból?

- A `button` és a `select`.
  - Egyes böngészőkben az `input` is.
- Az `img`-t a legtöbb böngésző `inline`-nak mutatja, de `inline-block`-szerűen viselkedik.
- Mindezt a HTML nem definiálja…
  - Egyszerűen a böngésző *alapértelmezett stíluslapjából* jön…

Mi tehet egy elemet „doboz”-szerűvé?

- A display módosítása az eddig tárgyalt módokon
  - `float`: `left` vagy `right`
  - `position`: `absolute` vagy `fixed`
- Tehát ha `float` vagy `position` állítást végzünk, általában fölösleges mellé még a display-t is bolygatni.
  - Egyrészt redundáns lesz tőle a kód.
  - Másrészt ok nélkül `display-t` állítani amúgy is törékenyebbé teszi a kódot.

Alapértelmezett stílusok

Miért fontos ez?

- Amiről eddig szó volt
  - A CSS „deklaratív” nyelv, ahol a kívánt hatást nem utasítások sorával érjük el, hanem tulajdonságok beállításával.
  - A tulajdonságok egyszerűnek tűnnek, de egymásra hatásukat kihasználva komplex komponenseket építhetünk (és ronthatunk el).
- Egy elem, amit stílusozni akarunk, sosem "üres"!
  - Mindig, minden CSS tulajdonság be van rajta állítva
    - vagy a CSS szerinti alapértelmezésre
    - vagy a böngésző alap stíluslapja szerint
    - vagy egy kollégád teljesen másik stíluslapja szerint…
- Bármilyen hatást akarunk elérni, bármilyen tulajdonságot akarunk beállítani, tudnunk kell, mit írunk felül.

Mit mond a spec?

„each User Agent (UA, often a 'web browser' or 'web client') will have a default style sheet that presents documents in a reasonable — but arguably mundane — manner.”

- A spec nem definiálja, mi az alapértelmezett stíluslap tartalma.
- Egyes böngészők – pl. a Chrome és az IE – saját, vendor-prefixes tulajdonságokat is bevezettek „saját használatra”
  - Így az alapértelmezett stíluslapról szabályozhatják a megjelenés olyan aspektusait, amit amúgy a CSS nem definiál.
  - Persze emiatt is nekünk fog fájni a fejünk később...

Reset CSS

Ötlet: ha a HTML dolga, hogy a tartalom szerkezetét írja le, akkor nem helyes, hogy bárminek is legyen alapértelmezett stílusa.

- kiemelni valamit nem csak félkövér betűvel lehet
- nem muszáj a linkeket aláhúzni
- ha a `h2` betűméretét túl nagynak találjuk, még véletlenül se intézzük el annyival, hogy "lefokozzuk" `h3`-ra

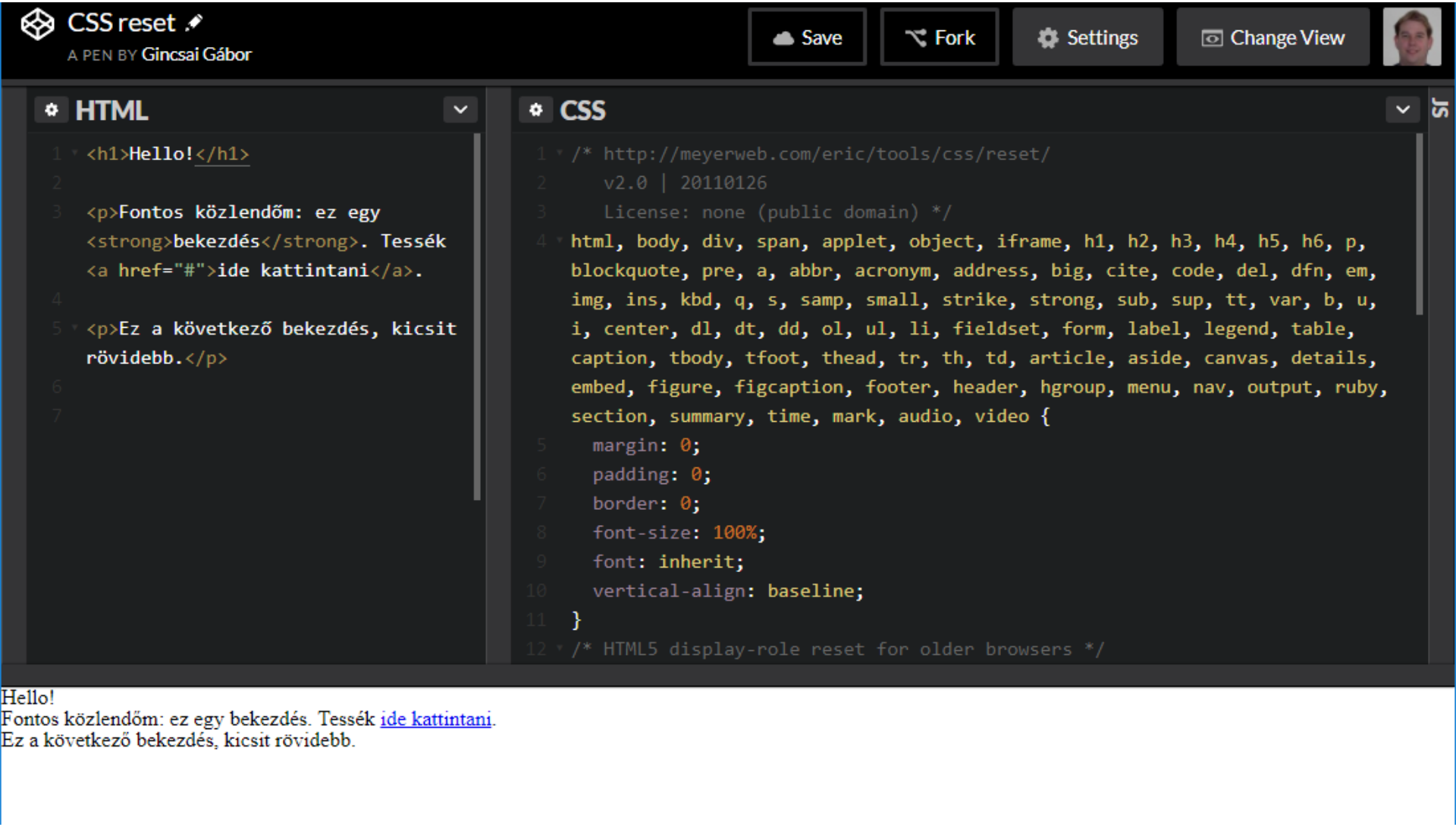
Reset CSS – mi a megoldás?

- Gyakorlatilag egy konyhakész stíluslap, ami szinte igyekszik mindent „kinullázni”.

```
html, body, div, h1, h2, h3, /* ... */ video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
```

- Főleg a margókat és a formázásokat
  - > de pl. linkeket nem.

Reset CSS



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

Reset CSS - értékelés

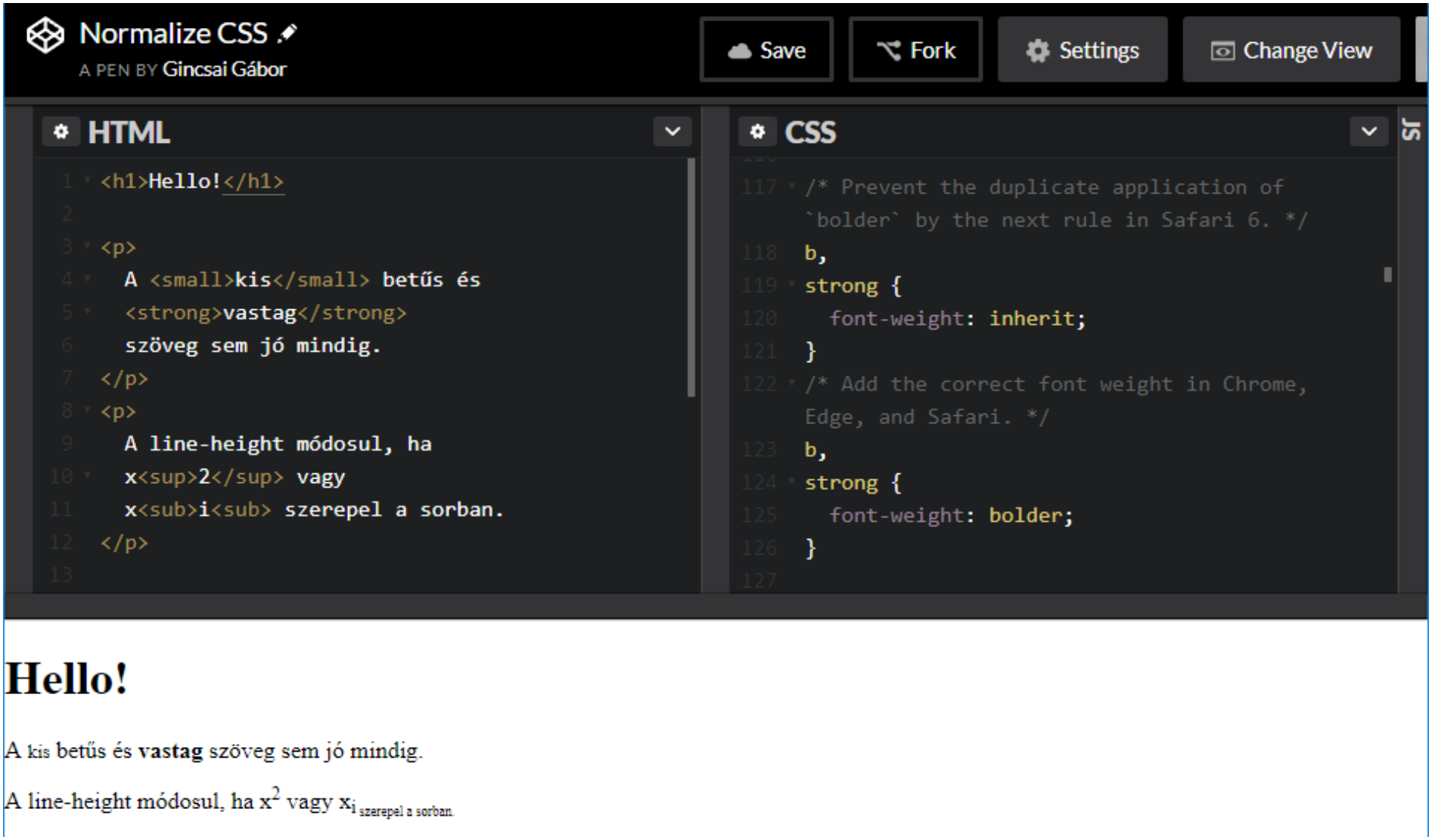
- Ami jó benne
  - Egy „üres vásznat” ad, ahol minden stílus, margó stb. tudatos döntésen alapul.
  - Nem kell azon agyalni, hol különböznek a böngészők alap stíluslapjai.
- Ami nem túl jó
  - A „Reset” szép név és koncepció, de nem igaz: felülírja a böngésző alap stílusait egy üres érzetet keltő stílussal, amit mi aztán újra felülírunk.
  - Ágyúval verébre: stílusokat aggat olyan elemekre is, amiket úgysem fogunk használni.

Normalize CSS

„When an element has different default styles in different browsers, normalize.css aims to make those styles consistent and in line with modern standards when possible.”

- Nem próbálja meg „kiütni” az alapértelmezéseket, csak elfedni a böngészők közötti eltéréseket.
- Általában nem „komponensként” használják (módosítatlanul benne hagyva a projektben, felülírva, ahol kell), hanem kiindulási alapként, ami a dizájn igényeknek megfelelően módosítanak

Normalize CSS



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

A stílus-kaszkád

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?

- Fontosság
- Származás
- Specifikusság
- Forrás sorrend

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 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 TFS taszkot, hogy ezt majd valaki csinálja meg rendesen
  - TODO komment nem ér.

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~~

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 0245-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

Forrás sorrend

- Ha két deklarációnak
  - Azonos súlyú a forrása (fontosság, származás).
  - Azonos a specifikussága.
- …akkor a forrás sorrend dönt.
  - Vagy is 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.

Tehát: mi alapján számítódik a súlyozás?

- Emlékeztetőül
  - Fontosság (!imporant)
  - Származás (böngésző alap, vagy saját CSS)
  - Specifikusság (pl: 0245 specifikusság)
  - Forrás sorrend (melyiket töltöttük be később)
- Ezalapján mindig tudni fogjuk, milyen érték állítódik be egy adott tulajdonságra?

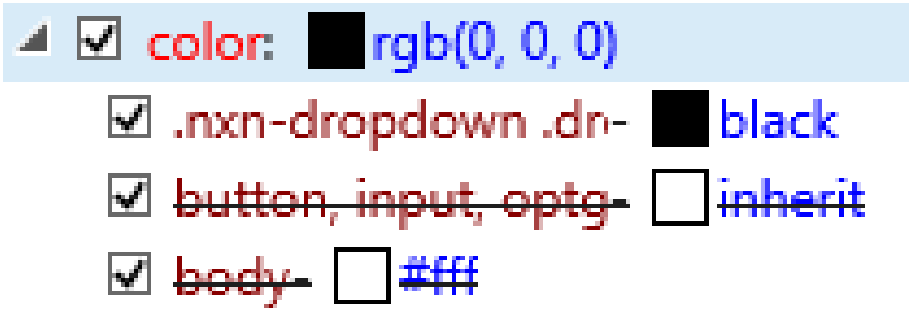

One more thing...

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)

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



A dokumentum folyam

Miről van szó?

- A HTML alapvetően egy dokumentumformátum.
- Ezért a HTML-ben szereplő tartalom alapvetően 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 tud egy sor CSS „elrontani” egy 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 */
}
```

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
  - ...

Mikor használjunk display: inline-t

A PEN BY Gincsai Gábor

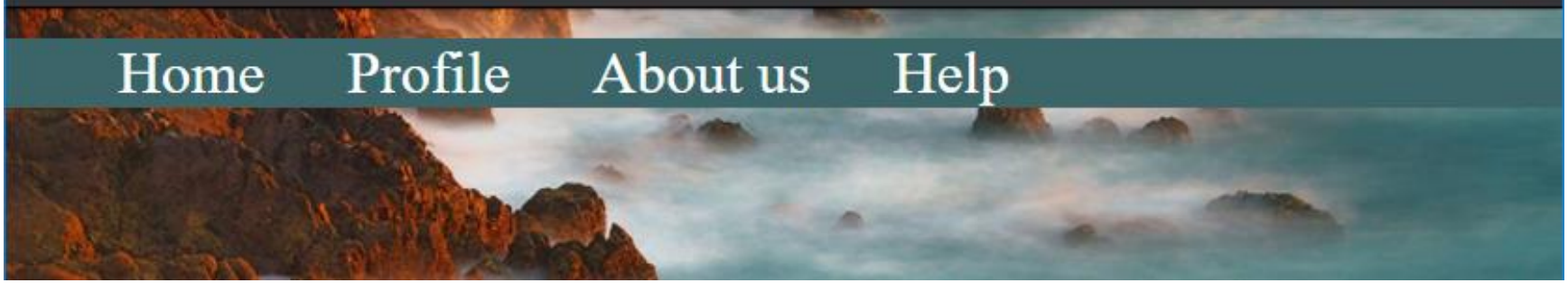
Save | Fork | Settings

HTML

```
<ul>
  <li>Home</li>
  <li>Profile</li>
  <li>About us</li>
  <li>Help</li>
</ul>
```

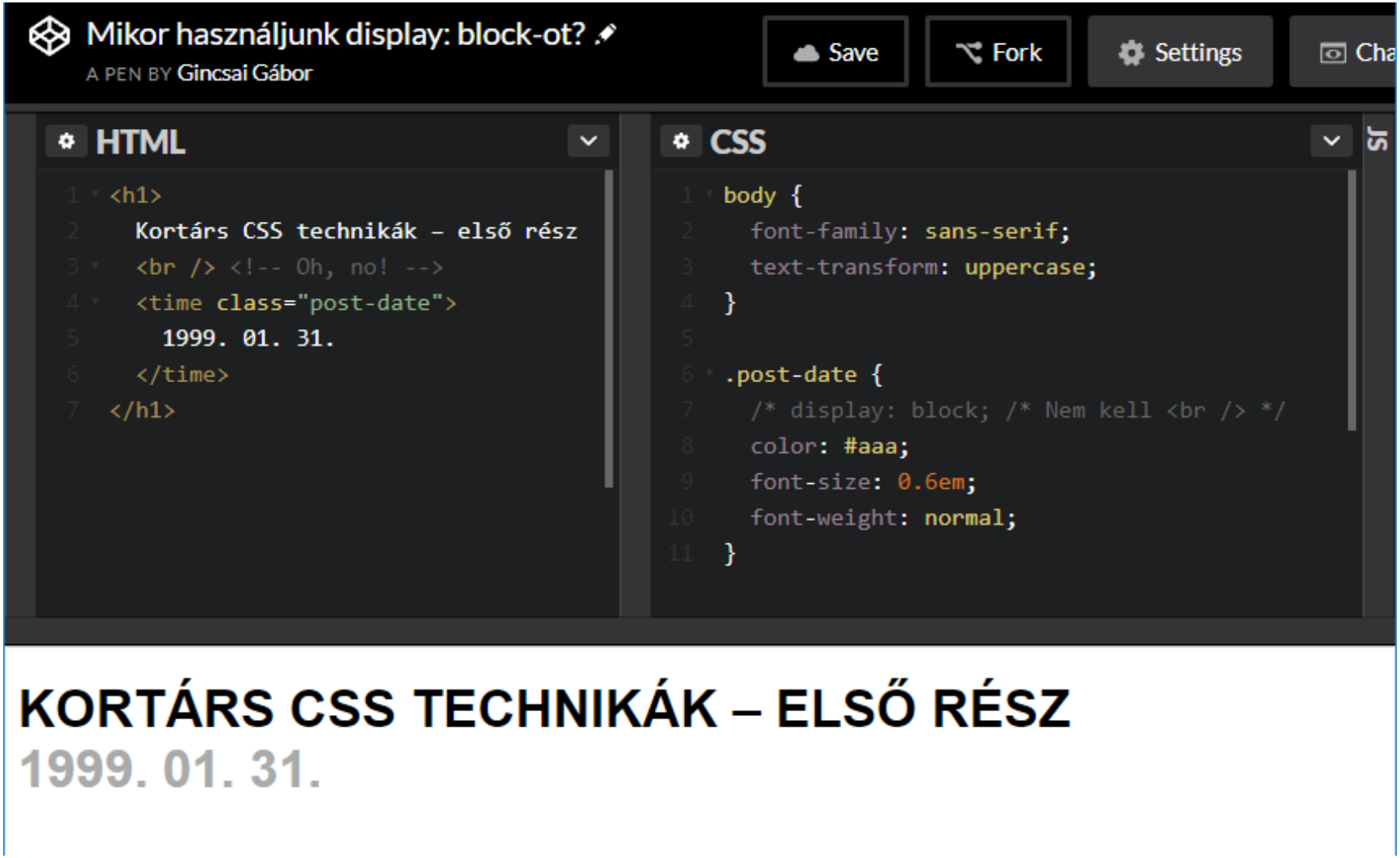
CSS

```
body {
  background-image: url(
'http://lorempixel.com/output/nature-q-c-1366-768-3.jpg');
  background-size: cover;
  margin: 0;
}

ul {
  background-color: #3C6569;
}

li {
  display: inline;
  color: white;
  font-size: 2em;
  padding: 20px;
```



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

Mikor jó a display: block?



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

A normál folyam megtörése I. – pozicionálás

- A `position` tulajdonság átállításával kivehetjük az elemet a normál folyamból, és általunk megadott koordináták mentén helyezhetjük el.
  - `position` deklarációnkat 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 folyam 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 :)

position: relative

A PEN BY Gincsai Gábor

Save | Fork | 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;
}
img {
  position: relative;
  left: 50px;
}
```

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

Hamburger ham hock pancetta
aola cupim, filet mignon short
cow turducken picanha salami.
cow hamburger pork chop jowl.
loaf tri-tip sirloin ham shank
uille prosciutto hamburger
k.

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

position: relative z-index

A PEN BY Gincsai Gábor

Save | Fork | 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/70/150/" />
</figure>
<figure>
  <img class="second" src="http://baconmockup.com/60/150/" />
</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.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 ribs beef ribs short loin venison porchetta fatback strip steak tongue drumstick.

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.

position: absolute

A PEN BY Gincsai Gábor

Save | Fork | 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

```
  width: 50%;
}
figure {
  float: left;
  margin: 5px;
  padding: 15px;
  background: pink;
}
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 salami. Beef cow hamburger pork chop ham shank andouille prosciutto

har

Pic f ground round short ribs beef ribs sho k strip steak tongue drumstick.

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.

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 dokumentummméretre működik jól…
- Fő hátrány: rugalmatlan
  - Főleg alpértelmezésben, 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)

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

HTML

```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

```css
body { background: #B8B9BB; }
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;
}
```

HTML

```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

```css
body { background: #B8B9BB; }
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;
}
```

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.

HTML

```html
<div class="page-wrap">
  <p>Bacon ipsum dolor amet pastrami boudin ribeye, andouille strip steak shank kevin bacon shankle. Boudin beef ribs prosciutto ham bresaola pig sausage tongue pork alcatra kielbasa strip steak. Strip steak meatloaf
```

CSS

```css
.page-wrap {
  width: 300px;
  margin: 0 auto;
  padding-top: 60px;
  /*transform: translate(0); */
}
h1 {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  margin: 0;
  padding: 20px;
  background: darkblue;
}
.green-container {
  /* transform: translate(0); */
  color: white;
  background: green;
}
```

Lorem ipsum

Bacon ipsum dolor amet pastrami boudin ribeye, andouille strip steak shank kevin bacon shankle. Boudin beef ribs prosciutto ham bresaola pig sausage tongue pork alcatra kielbasa strip steak. Strip steak meatloaf capicola, drumstick picanha pork belly jowl jerky spare ribs. Cupim beef t-bone, tail landjaeger corned beef doner pig. Pork chop rump fatback, strip steak andouille pork loin meatball turkey. Bresaola pig turkey hamburger, tail short loin swine pork belly. Pork chop filet mignon pork belly spare ribs ball tip boudin biltong meatball meatloaf pancetta fatback short ribs.

BLABLABLA

Picanha pork chop shank, meatloaf ground round short ribs beef ribs short loin venison porchetta fatback strip steak tongue drumstick. Salami beef meatloaf fatback. Porchetta turkey tri-tip ground round. Frankfurter ground round cow biltong bresaola, salami corned beef pork belly ham hock kielbasa drumstick pork loin. Brisket shoulder flank, corned beef chicken kevin short ribs rump pork loin pork chop. Picanha pig beef ribs tenderloin shoulder shank. Jerky beef leberkas, tri-tip shoulder picanha turkey sausage kevin boudin pork loin pork chop.

Burgdoggen leberkas beef ribs, meatball

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.

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;
}
*/
```

Stacking context

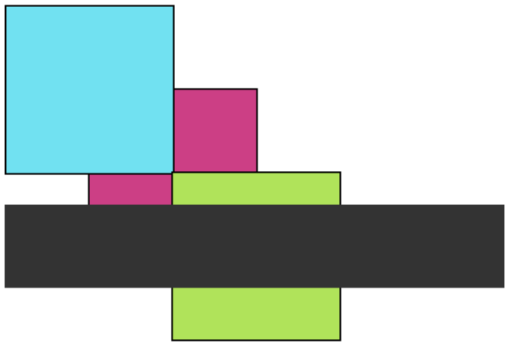
- A z-index egyszerű is lehetne, de nem az
  - Az ok: nem globális számról van szó, a szülők (és egyéb felmenők) értékei is számítanak!
- Pontosabban: a z-index mindig az aktuális stacking contexten belül érvényesül, így „globálisan” nem kerülhet alacsonyabb vagy magasabb rétegre, mint a stacking context 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

HTML

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

CSS (LESS)

```less
.container {
  position:relative;
  z-index:3;  /* minden benne léve 3-asak */
}
section {
  position:relative;
  top:120px;
  z-index:5;     /* ez van felül, container: 3 ez 5 */
  background:#333;
  width:300px;
  height:50px;
}
.pink {
  top:50px;
  left:50px;
  background:#cc3f85;
  z-index:1;  /* containeren belüli sorrend */
}
.blue {
  background:#71e1f1;
  z-index:6; /* containeren belüli sorrend */
}
.green {
  top:100px;
  left:100px;
  background:#b0e35a;
```



A normál folyam megtörése II. – float

- 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

float image

A PEN BY Gincsai Gábor

Save | Fork | Settings

HTML

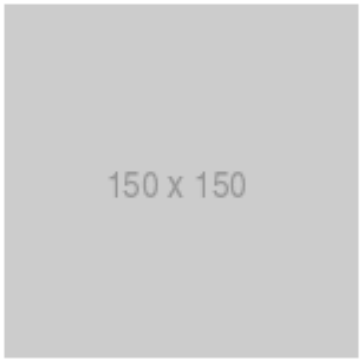
```
<article>
  <header>
    <h1>Dyin' ain't much of a livin', boy</h1>
  </header>
  <main>
    <img src="http://placehold.it/150x150" />
    <p> I don't think they tried to market it to the billionaire, spelunking, BASE-jumping crowd. Well, what is it today? More spelunking? What you have to ask yourself 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
```

CSS (LESS)

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

JS

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.

A float használata layoutra

- Layoutra a `<table>` alapú layout idejétmúlttá kikiáltását követően kezdték el használni, és ma is ez a domináns megoldás.
  - Ez "hajtja" pl. a Bootstrapet és a Foundationt is.
- Valószínűleg ez így is marad, amíg a flexboxot kellően elterjedtnek nem nyilvánítjuk.
  - A Bootstrap 4 már flexbox alapú!

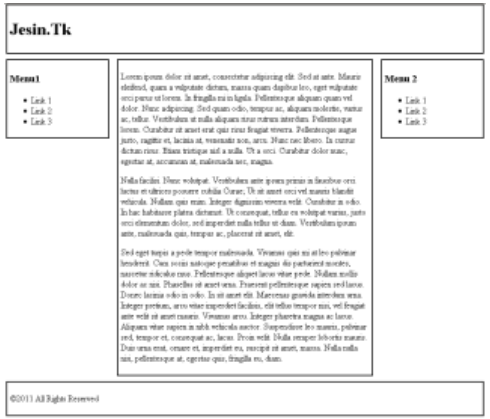
A float hatása a dokumentumfolyamra

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

Hagyományos layout megoldások

Float blokkok halmozódása

- Ha két elemet is ugyanabba az irányba „floatolunk”, *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



Clear property

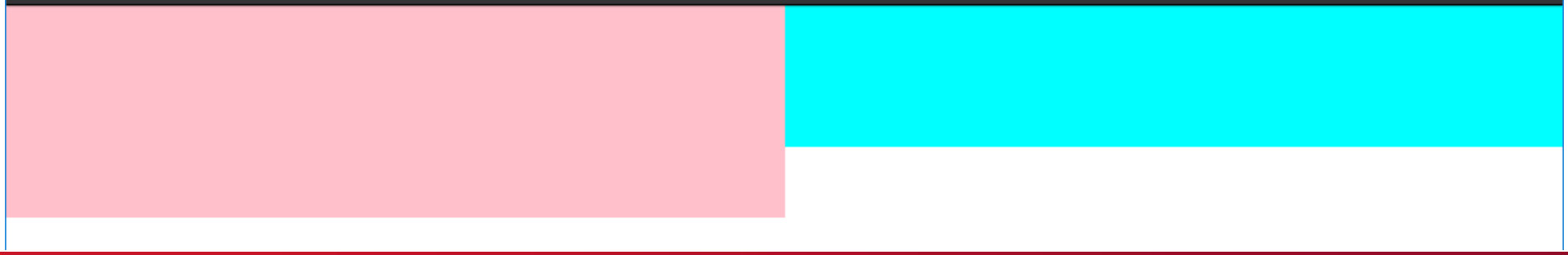
- A float arra való, hogy körbefuttathatóvá tegyen dolgokat.
  - A nem-floatolt elemek szépen körbe is futják, ha módjuk van rá.
  - De így például a footer nem a lap alján van, hanem „beszorul” a két panel közé.
- Ennek megodására való a `clear` property
  - Letiltja a körbefuttatást egyik vagy másik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

HTML

```html
<main class="column1"></main>
<aside class="column2"></aside>
```

CSS (LESS)

```less
body{ margin: 0 }
.column1 {
 float:  left;
 width: 50%;
 min-height: 150px;
 background:  pink;
 /*margin: 0 2.5%; /* width legyen 45% */
 /* margin-left: 3.3%; */
}
.column2 {
 float: right;
 width: 50%;
 /*margin: 0 2.5%; /* width legyen 45% */
 min-height: 100px;
 /* margin-left: 3.3%; /* legyen ez is float: left */
 background:  cyan;
}
/*
@media all and (max-width: 400px) {
    .column1, .column2 {
        width: 95%;
        margin: 10px 5%;
    }
}*/
```



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

float layout basics multicolumn

A PEN BY Gincsai Gábor

Save | Fork | Settings | Cha

HTML

```
<header class="col-100">header</header>
<main class="col-66 col-m-100">main</main>
<aside class="col-33 col-m-100">aside</aside>
<footer class="col-100">footer</footer>
```

CSS (LESS)

```
[class*="col-"] {
  float: left;
  background: gold;
  border: 1px solid;
  min-height: 40px;
  margin: 0 0 5px;
}

.col-100 { width: 100%; }
.col-66 { width: 66.67%; }
.col-33 { width: 33.33%; }

@media all and (max-width: 600px) {
    .col-m-100 { width: 100%; }
    .col-m-66 { width: 66.67%; }
    .col-m-33 { width: 33.33%; }
}
```

header

main | aside

footer

HTML

```html
<main class="column1"></main>
<aside class="column2"></aside>
<footer>footer</footer>
```

CSS

```css
.column1 {
 float: left;
 width: 45%;
 min-height: 150px;
 background: pink;
}
.column2 {
 float: right;
 width: 45%;
 min-height: 100px;
 background: cyan;
}
footer {
  background-color: lightgreen;
  min-height: 80px;
  /* clear: both; */
```

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

HTML

```html
<div class="container">
  <main class="column1"></main>
  <aside class="column2"></aside>
  <!-- <div style="clear:both;"></div>-->
</div>
```

CSS (LESS)

```css
.container {
  background-color: darkgray;
  /* border: 5px solid; */
  /* overflow: hidden; */
}
/*.container:after {
    content: " ";
    display: block;
    clear: both;
}*/
.column1 {
 float: left;
 width: 45%;
 min-height: 100px;
 background: pink;
}
.column2 {
 float: right;
 width: 45%;
 min-height: 50px;
 background: cyan;
}
```

JS

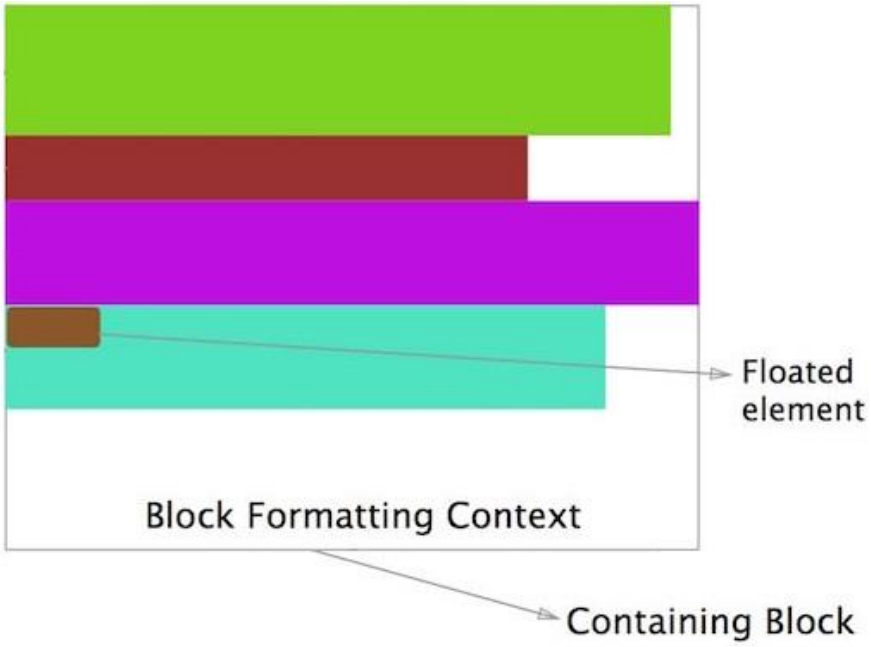
https://codepen.io/gincsai/pen/JrwBBy

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

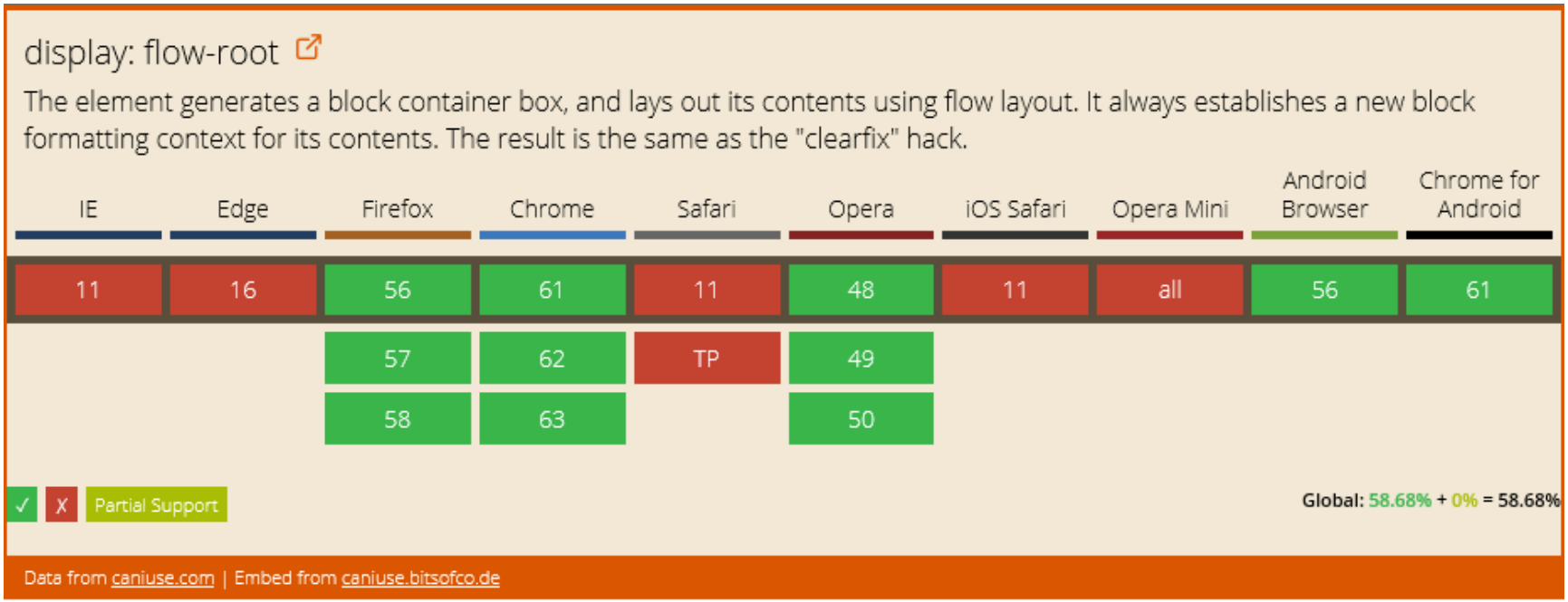


Mi definiál új block formatting contextet?

- Először is: a gyökérelem (`<html>`)
- A floatolt elemek (ahol a `float` nem **`none`**)
- Az abszolút vagy fixen pozicionált 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 gond, hogy a példában **a container nem volt önálló block formatting content**, de az overflow: hidden azt csinált belőle.

Nincs olyan property, ami semmi mást nem csinál, csak új block formatting contextet hoz létre?

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



Az inline-block alapú layout

- Nem annyira elterjedt, de egész jól működő alternatíva blokkok egymás mellé helyezésére.
- Előnyei:
  - Nem kell `clear`, sem clearfix.
  - Egymáshoz képest igazíthatóak `vertical-align`-nal.
  - A hatása a dokumentumfolyamra minimális, kevésbé törékeny.
- Amire figyelni kell:
  - Szélességüket alapvetően a tartalmuk határozza meg (ellentétben a normál blokkokkal).
  - Ezért általában `width` vagy `min-width` megadása is szokott szerepelni, ha pl. panelekre használjuk.

HTML

```html
<div class="container">
  <main class="column1"></main>
  <aside class="column2"></aside>
</div>
<footer>footer</footer>
```

CSS (LESS)

```less
.container {
  font-size: 0;
}
.column1,
.column2 {
  display: inline-block;
  width: 48%;
  font-size: 1rem;
  vertical-align: top;
  margin: 0 1%;
}
.column1 {
  min-height: 100px;
  background: pink;
}
.column2 {
  min-height: 50px;
  background: cyan;
}
footer {
  background-color: lightgreen;
  min-height: 80px;
}
```

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

HTML

```
<div class="gallery">
  <div class="gallery-item">
    <img src="http://baconmockup.com/140/120/">
    <h2>title</h2>
  </div>
  <div class="gallery-item">
    <img src="http://baconmockup.com/140/50/">
    <h2>title</h2>
  </div>
  <div class="gallery-item">
```

CSS (LESS)

```
body {
  text-align:center;
}
.gallery-item {
  /*float:left; /* csúnya */
  display: inline-block;
}
```

title title title title title

title title title

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

A display: table-* alapú layout

- Wait, what? A táblázat alapú layout rossz dolog, nem?
  - Ha `<table>`, `<tr>`, `<td>` tagekkel oldjuk meg, akkor tényleg az.
- Miért szerettük régen a táblázat alapú layoutot?
  - Teljes(ebb) kontroll az elemek elhelyezkedésén.
    - Méretezés, dolgok egymás mellé helyezése.
  - Ugyanakkor lényegesen rugalmasabb, mint a `position`, alkalmazkodik a tartalom méretéhez.
  - `vertical-align`-ra reagál.

A <table> alapú layout problémái…

- Reszponzív / mobil dizájn nem oldható meg vele.
- Az oldal folyószöveggé alakítását (pl. felolvasók számára) ellehetetleníti.
- Erős csatolás a tartalom és a megjelenés között.
- „Szemantikailag” teljesen helytelen.

Viszont a `float`, amit helyette használunk, bonyolultabb és törékenyebb…

Mi is az a display: table-*?

- A `table-*` propertyk eredeti célja, hogy pusztán CSS-ből reprodukálhassuk a HTML `<table>`, `<td>` stb. elemek viselkedését
  - Pl. ha tényleg táblázatba való adatokhoz használjuk, megoldható vele az, hogy csak desktopon jelenjen meg igazi táblázatként, és mobilon inkább csak lista legyen
- A lehetőség, hogy tiszta CSS-ből táblázatot csinálhatunk, a lehető legjobb kombinációnak tűnik
  - a táblázat alapú layout kényelme és megbízhatósága
  - a HTML-be drótozás kellemetlenségei nélkül

table-* értékek

Display property	HTML megfelelője
`table, inline-table`	`table`
`table-row`	`tr`
`table-cell`	`td`
`table-row-group`	`tbody`
`table-header-group`	`thead`
`table-footer-group`	`tfoot`

Rendeltetés szerű használat

HTML

```
<div class="table">
  <div class="table-row">
    <div class="table-head">Table Header</div>
    <div class="table-head">Table Header</div>
    <div class="table-head">Table Header</div>
  </div>
  <div class="table-row">
    <div class="table-cell">Table Cell</div>
    <div class="table-cell">Table Cell</div>
    <div class="table-cell">Table Cell</div>
  </div>
    <div class="table-row">
    <div class="table-cell">Table Cell</div>
    <div class="table-cell">Table Cell</div>
    <div class="table-cell">Table Cell</div>
  </div>
</div>
```

CSS (LESS)

```
.table{
  display: table;
  width:100%;
}
.table-row{
  display: table-row;
}
.table-cell, .table-head{
  display: table-cell;
  padding:1em;
  border: 1px solid red;
}
.table-head{
  font-weight:bold;
}
```

Table Header	Table Header	Table Header
Table Cell	Table Cell	Table Cell
Table Cell	Table Cell	Table Cell

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

Függőleges igazítás

Függőleges igazítás (table)

A PEN BY Gincsai Gábor

Save | Fork | Settings | Cha

HTML

```
<div class="wrap">
  <div class="block">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
  </div>
  <div class="block">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit.
  </div>
</div>
```

CSS (LESS)

```
body{
  font-family:sans-serif;
}
.wrap{
  display: table;
  max-width:900px;
  margin:auto;
}
.block{
  display: table-cell;
  vertical-align: middle;
  width: 50%;
  border: 1px solid red;
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

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

Többoszlopos elrendezés

HTML

```
<div class="grid">
  <div class="col">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
  </div>
  <div class="col">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit.
  </div>
    <div class="col">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
```

CSS (LESS)

```
* { box-sizing: border-box; }
body{
  font-family:sans-serif;
}
.grid{
  display: table;
  width: 100%;
  border-spacing:.5em;
}
.col{
  padding:1em;
  width:25%;
  display:table-cell;
  border: 1px solid red;
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

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

Navbar

Navbar (talbe)

A PEN BY Gincsai Gábor

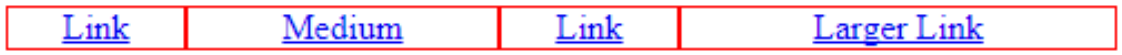
Save | Fork | Settings

HTML

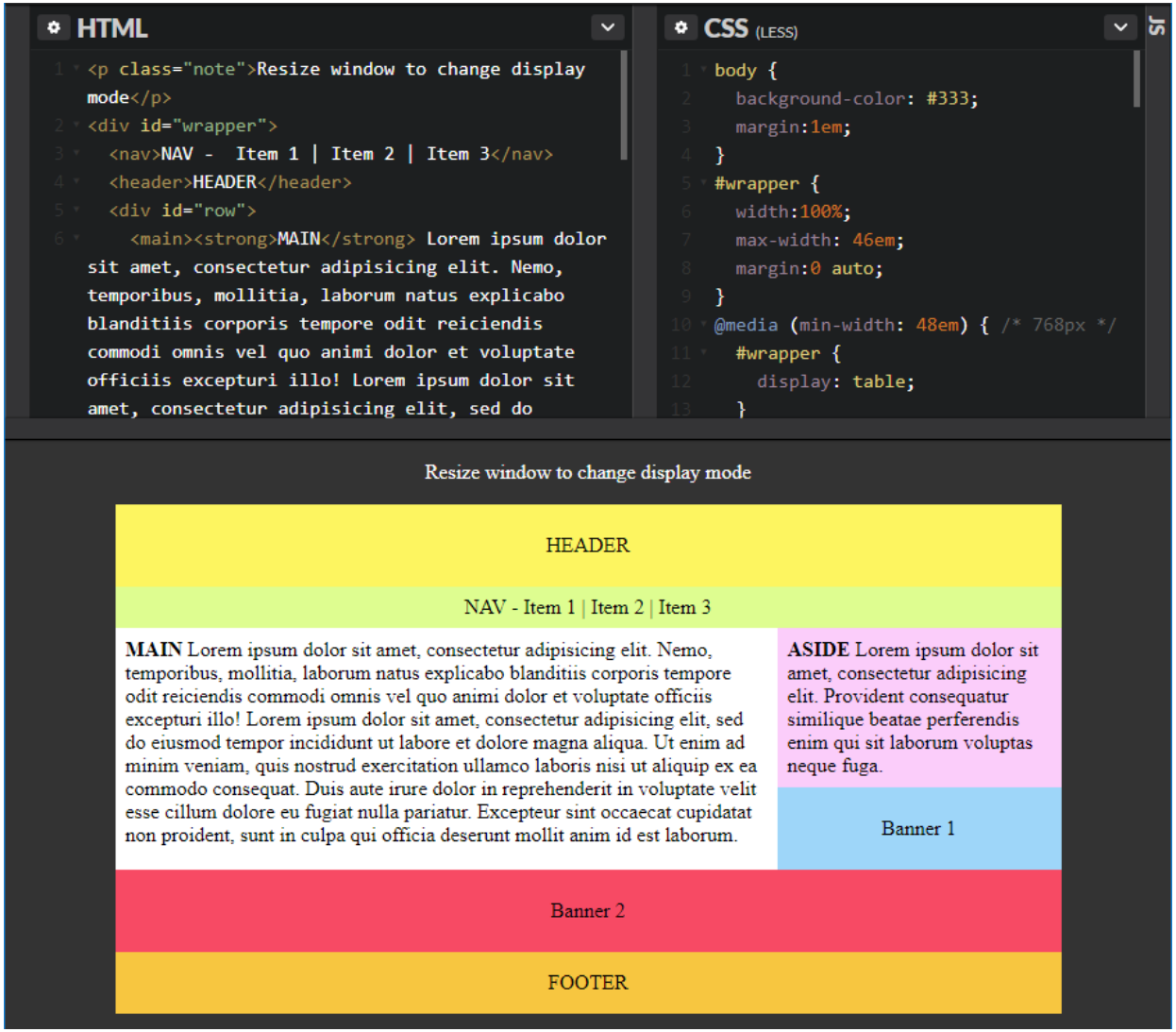
```
<ul class="navbar">
    <li><a href="#">Link</a></li>
    <li><a href="#">Medium</a></li>
    <li><a href="#">Link</a></li>
    <li><a href="#">Larger Link</a></li>
</ul>
```

CSS (LESS)

```
.navbar {
    width:500px;
    display:table;
}
.navbar li {
    display: table-cell;
    text-align: center;
    border: 1px solid red;
}
```



Reszponzív layout



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