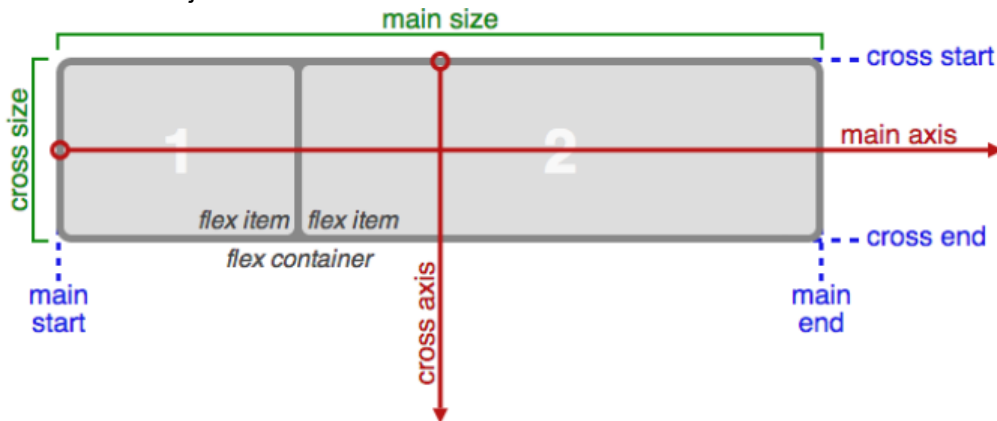


WEB03-Flexbox, Bootstrap, SCSS

Flexbox

A **flexbox** teljesebb kontrollt ad egy konténer gyerekei felett, *reszponzív* módon

- Elemek igazítása egymáshoz és a konténer széleihez
- Elemek automatikus méretezése arányosan
- Elemek sorrendjének variálása



Tulajdonságok a konténeren

`display: flex`: Az elemeket befoglaló konténeren meg kell adni, hogy flexbox elrendezést használjon.

`flex-direction`: Gyerekek elrendezése

- `row`: vízszintesen

- `column`: függőleges tengely mentén

`flex-wrap`: sor/oszloptörés engedélyezése

`justify-content`: Elemek **elrendezése**, tagolása a főtengety (*main axis*) mentén

- A gyerekek közt megmaradt helyet osztja be.

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



`align-items`: Gyerekek igazítása a **főtengelyre merőlegesen**

! Hasonlít a `justify-content`-re, de fontos különbség, hogy **nem a gyerekek közt 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 főtengety mentén nyúlik).


- A gyerekek a kereszttengetylen is nyújthatók, erre van a `stretch` opció

- Például ha 3 különböző magasságú flex konténerünk van, akkor az `align-items: center;` a kereszttengety mentén középre rendezi őket.

`align-content`: **Több sornyi flexbox elemek sorainak** igazítása a kereszttengety mentén.

- Egyetlen sornyi flexbox elemre nincs hatása.

- Értékei kb. ugyanazok és kb. ugyanazt jelentik, mint a `justify-content` esetén, azonban itt van külön `stretch` érték is (ami a default), itt a gyerekektől függ a „nyúlás”.

 `align-items` és `justify-content`: egyes elemekre vonatkoznak.

 `align-content`: Több sornyi elemre vonatkozik, a kereszt tengely mentén

Tulajdonságok a konkrét flex elemeken

`flex-grow`: szabályozza, hogy az elem nyúljon-e, ha marad hely a fő tengelyen

`flex-shrink`: összenyomható-e az elem, ha kevés a hely, és nem lehet sort törni.

`flex-basis`: az elem alapértelmezett mérete, nyújtás/összenyomás előtt

- alpból ez is `auto`, ami a gyerek fő tengely menti méretét veszi fel

- Megadhatunk konkrét értéket is, ekkor *felülírjuk a fő tengely menti méretet.*

`flex`: shorthand az előző háromra

`flex: 1 1 auto`: rugalmas

`flex: 0 0 auto`: rugalmatlan

`align-self`: az elem felülírhatja saját magára a konténerek beállított `align-items` értéket.

`order`: Az elem alapvetően a HTML-ben definiált sorrendben jelenik meg. De ezzel variálhatunk rajta. **Negatív szám is lehet, alapértelmezetten 0.**

 Nem alkalmazható az egyes flexbox elemekre a: `float`, `clear`, `vertical-align`

Bootstrap

- 👍 Könnyen tanulható és használható a részletes dokumentáció miatt.
- 👍 Kiváló grid rendszer a reszponzív layout kialakításához.
- 👍 Nem kell mindent nulláról megírni, csomó kész stílussal érkezik.
- 👍 Rengeteg időt meg lehet vele takarítani.
- 👍 Számos shortcut osztályt tartalmaz (pl. `success`, `warning`, passzoló színekkel és ikonokkal)
- 👍 Sok segítséget ad az űrlapok és táblázatok formázásához.
- 📘 Kiindulási alapnak használjuk

Bootstrap Grid

mobile first tervezés: mobilra mondjuk meg az elrendezést, majd ezt a képernyő növekedésével fokozatosan felülbírálhatjuk.

- Sorokra és oszlopokra bontja a képernyőt
 - Minden sor **12** azonos méretű oszlopból áll
 - A sorok száma, mérete tetszőleges.
 - Egy konkrét tartalmi elem több oszlopon is átnyúlhat
 - A sorok magasságát jellemzően a legmagasabb konkrét cellája határozza meg.

Használata

A grid gyökere a `container` vagy `container-fluid` osztály

- Ebbe pedig `row` osztállyal ellátott konténereket teszünk, amikbe kizárólag `col-*` osztályú elemek kerülnek.

✏️ A col-* osztályok viselkedése

Egy prefixszel megadhatjuk, melyik töréspont **fölött** alkalmazza

- pl: `col-sm-*` a tableteken és desktopokon érvényesül, de mobilon nem

A specifikusabb osztály felülírja a kevésbé specifikusat!

- `xs >> sm >> md >> lg >> xl >> xxl`

- `*`: megadja, hogy hány oszlopot foglaljon el a cella, Alapértelmezetten: **12**
- A reszponzív oszlopszélességekhez a különböző töréspontokra vonatkozó oszlop szélességeket egyszerre kell megadni. Vagyis egy-egy elemnek több class-t kell megadni. **pl egyszerre** `.col-md-4` és `.col-6`
 - A `col-{méret}` **minden felbontásra vonatkozik**, hiszen ez a legkisebb.
 - A `col-md-{méret}` viszont **csak 768 px felett**.

A Bootstrap 4-től a `col`-t ímegában s használhatjuk mint lehetséges oszlop szélességet.

- Ha nem adjuk meg a töréspontot, akkor ebben az esetben az adott oszlop **mérete automatikusan kerül meghatározásra**.

`col-{méret}-auto`: Lehetőség nyílik változó szélességű oszlopok definiálására is.

- A **tartalom mérete határozza meg** az oszlop szélességét.

⚠️ Egy cellán belül vehetünk fel új `row`-t amit a `col`-ok segítségével további cellákra oszthatunk.

`.row-cols-{sm}-*` osztály segítségével gyorsan **megadhatjuk, hogy mennyi oszlopot szeretnénk használni**

- A gyerekekre így már csak a `.col` osztályt kell rátennünk!

- ❗ ez csak egy egyszerűsített megadása a `.col-*` osztályoknak, amik rákerülnek az egyes oszlopokra.
- A `.row-cols-auto` is használható.

Bootstrap osztályok flexboxos elrendezéshez

Bootstrap	Pure CSS
<code>.d-{sm}-flex</code>	<code>{ display: flex }</code>
<code>.flex-{sm}-column</code>	<code>{ flex-direction: flex-column }</code>
<code>.justify-content-{sm}-between</code>	<code>{ justify-content: space-between }</code>
<code>.align-items-{sm}-center</code>	<code>{ align-items: center }</code>
<code>.align-self-{sm}-center</code>	<code>{ align-self: center }</code>
<code>.flex-{sm}-grow-1</code>	<code>{ flex-grow: 1 }</code>
<code>.flex-wrap</code>	sortörés engedélyezése
<code>.m{se}-auto</code>	jobbra / balra igazítás automatikus margóval.
<code>.m{t b r}-{0-9}</code>	<i>felső/alsó/bal/jobb</i> margó
<code>.p{t b r}-{0-9}</code>	<i>felső/alsó/bal/jobb</i> padding

- 🔖 Az oszlopokat **függőlegesen is tudjuk igazítani** az alábbi bootstrap classokkal
- `align-self-start`, `align-self-center`, `align-self-end`
- 🔖 A teljes sorokat is lehet függőlegesen igazítani, `align-items`-el

HTML elemek reszponzív megjelenítése / elrejtése

`d-{value}` mérettől függően mindig elrejt/megjeleníti

`d-{breakpoint}-{value}`: adott képernyő méreten elrejt/megjeleníti

`{value}`: a lehetséges display értékek: `non`, `inline`, `inline-block`, `block`, `flex`, ...

- ❓ Annyira sok formázást tartalmaz a bootstrap, hogy az összes osztályt nem lehet felsorolni. De van hozzá jó dokumentáció: <https://getbootstrap.com/docs/5.2/getting-started/introduction/>

Cheat sheet: <https://getbootstrap.com/docs/5.0/examples/cheatsheet/>

SCSS

A **SASS** egy CSS preprocesszor. Új funkciókkal bővíti és egyszerűsíti az életünket:

- változók
- szabályok egymásba ágyazása
- mixinek
- függvények és operátorok

A végén az elkészített fájlból CSS-t állítunk elő, így a böngészőhöz ugyanúgy CSS jut el, *csak hatékonyabban tudjuk elkészíteni*.

A SASS preprocesszornak kétféle szintaxisa van: SASS és SCSS. A tárgy csak az SCSS-el foglalkozik.

Változók

- Értékek újrahasznosítása, úgy, hogy csak egy helyen kelljen megadni.
- Szintén van scope-juk.

👍 Átláthatóbb kód

👍 Könnyebb karbantartani, mert csak egyetlen helyen kell lecserélni.

```
$page-min-width: 1200px;
$page-min-height: 800px;

html {
  min-width: $page-min-width;
  min-height: $page-min-height;
  height: 100%;
}
```

példa változók használatára

```
$text-color: blue; // globális változó

.error {
  $text-color: red; // lokális változó
  color: $text-color;
}

.normal-text {
  color: $text-color;
}
```

változók lokalitása

```
$text-color: blue;
.error {
  $text-color: red;
  color: $text-color;
  $text-color: green !global;
}

.normal-text {
```

```
color: $text-color; //zöld lesz
}
```

!global flag.. Ilyenkor a névütközés esetén a külső, globális változóra hivatkozik az SCSS.

Mixinek

A mixinekkel egy **osztályba szervezzünk több tulajdonság beállítását**, amit később egy sorban újra **hasznosítunk**, akár úgy is, hogy *bemenő paramétert* kap.

Hasznos lehet a vendor prefixek miatti többféle szabálybeállításnál.

```
@mixin square($size, $color) {
  width: $size;
  height: $size;
  background-color: $color;
}

.small-blue-square {
  @include square(20px, rgb(0,0,255));
}

.big-red-square {
  @include square(300px, rgb(255,0,0));
}
```

mixinek. Újra felhasználunk egy kódrészletet.

Függvények

```
@function column-width($col, $total:8) {
  @return percentage($col/$total);
}

.col-3 {
  width: column-width($total: 8, $col: 3); // 37.5%
}
```

Függvények létrehozásánál az egyes paramétereknek alapértelmezettértéket is adhatunk. Ha több paramétere van egy függvénynek, akkor a nevesített paraméterekkel átláthatóbb kódot készíthetünk és a paraméter sorrendet sem kell betartani.

Szabályokat generálhatunk **ciklusok** segítségével is:

```
@for $i from 1 through $total {
  .col-#{ $i } { width: column-width($i) };
}
```

Mixin vagy függvény?

A Mixin kimenete közvetlenül CSS-re fordítható. Tulajdonság érték párokat kapunk vissza. A függvények viszont egy konkrét értéket adnak vissza, amit egy CSS tulajdonsághoz kell rendelni.

Vagyis **akkor használjunk függvényt, ha csak számításokat szeretnénk végrehajtani.**

Egymásba ágyazás:

Sokkal átláthatóbban fogalmazhatjuk meg az összetett CSS szabályokat, ha azokat egymásba ágyazzuk:

```
ul {  
  list-style: none;  
  li {  
    padding: 15px;  
    display: inline-block;  
    a {  
      text-decoration: none;  
      font-size: 16px;  
      color: #444;  
    }  
  }  
}
```

Parent selector (&):

A **&** lehetőséget ad összetettebb szabályok definiálására is, a gyerekből meaghatunk a szülőre vonatkozó szabályokat is. (hasznos pl. hover esetén)

Extend

SCSS-ben az **extend** segítségével tudjuk megoldani azt, hogy a *több szabályban is használt részleteket egy helyre szervezzük ki és később azt az új szabályokban csak kiegészítsük.*

Extend vs. Mixin

Fontos különbség a mixinhez képest, hogy itt **az extend ezeket a szabályokat egyetlen css blokkba generálja.**

- *Pl.* ha a `.message` és a `.success`-be is betettük az extend-et, akkor az egy `.messgae, .success {` fejlécű blokkot tesz a css-be, míg a mixin belegenerálja külön mindkettőnek a blokkjába az értékeket.

Importálás

Nagy méretű SCSS esetén átláthatóbbá tehetjük a kódot, ha azokat külön fájlokba szervezzük. Importálásra használjuk az **@import** kulcsszót.

Gincsei Gábor diasorai alapján.