# Flexbox 2.

## Bevezetés

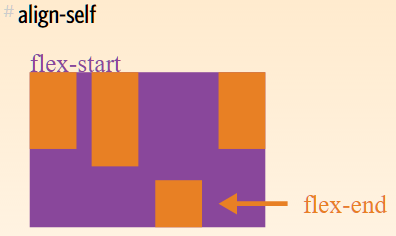
Az előző leckében csak a flex container tulajdonságait állítottuk be. Most megnézzük, hogy milyen beállítási lehetőségei vannak az egyes flex elemeknek.

## Előkészületek

1. Nyisd meg a *dobozok* mappát, majd abból a *doboz.html és a doboz.css* fájlokat a Visual Studio Code-ban!
2. Nyisd meg az oldalt a böngészőben is!
3. Nézd át az eddig elkészített HTML és CSS kódokat!

## Elem igazítása külön

A flexbox container elemnél az align-items tulajdonsággal megadhatjuk, hogy az elemek a fő irányra merőlegesen hogyan legyenek igazítva. Ezt az egyes elemeknél az align-self tulajdonsággal felülbírálhatjuk. Ugyanazok a lehetőségek vannak, mint az align-items tulajdonságnál.

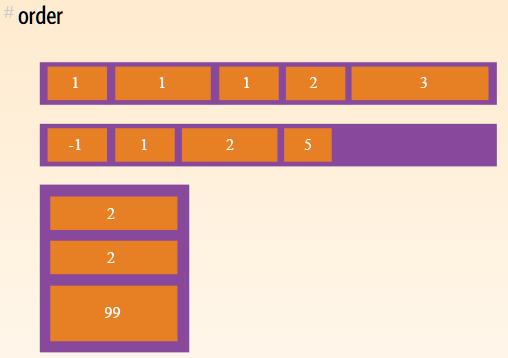


1. Függőleges irányban most minden elem középre van igazítva. Állítsd be, hogy a kettes doboz függőlegesen szét legyen húzva a teljes magasságra, a négyes pedig alul legyen!  
   
2. Próbáld ki a többi lehetőséget is, majd töröld a külön igazításokat!

## Megjelenítés sorrendje

Alaphelyzetben az elemek a HTML fájl szerinti sorrendben jelennek meg. Ezt a flexbox alkalmazásával módosíthatjuk, ami különösen reszponzív weblapoknál hasznos.

Minden elemhez hozzárendelhetünk egy számot az order tulajdonsággal. (Alaphelyzetben mindegyikhez 0 van hozzárendelve.) A kisebb számúakat előbb, a nagyobbakat később jeleníti meg. Ugyanakkora számoknál a HTML-beli sorrend marad.



1. Állítsd be, hogy a 2-es elem a végén, a négyes pedig az elején jelenjen meg!  
   
2. Próbáld ki, majd állítsd vissza az eredeti sorrendet!

## Szabad hely elosztása

Az eddigiekben a dobozok szélessége nem változott. A flex tulajdonság segítségével viszont rugalmassá tehető a dobozok szélessége, és így könnyen készíthető reszponzív weboldal is.

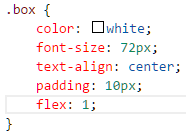
Ha van a sorban vagy oszlopban a dobozokon kívül üres hely, akkor azt eloszthatjuk a dobozok között.

A legegyszerűbb esetben a flex tulajdonságnál csak egy számot adunk meg (flex-grow). Ez azt jelzi, hogy a rendelkezésre álló üres helyből az adott doboz milyen arányban részesedjen.

Például állítsuk a box4 doboz flex tulajdonságát 1-re:  


Mivel a többi elemnél nem adtunk meg értéket, és az alapértelmezett érték 0, az összes szabad helyet a 4-es doboz kapta meg.

Töröljük ez előző beállítást, és állítsuk most minden doboz flex tulajdonságát 1-re:



Most azt tapasztaljuk, hogy minden doboz szélessége ugyanakkora lett, és teljesen kitöltik a sort.

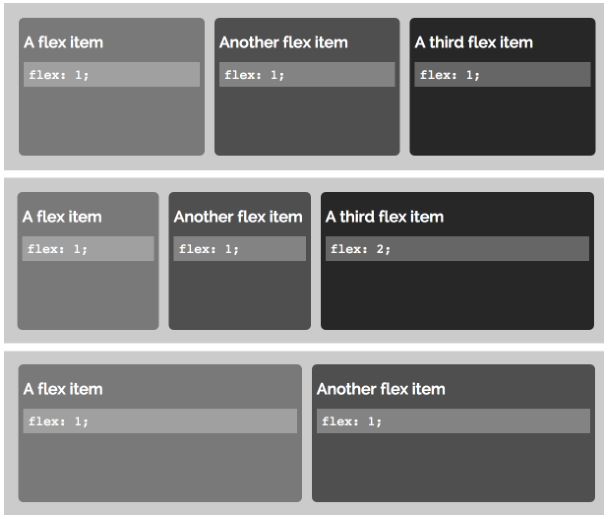
Állítsuk most az 1-es doboz flex tulajdonságát 2-re:



Így most az első doboz kétszer olyan széles lett, mint a többi.

Vajon mi történne, ha minden doboz flex értékét 2-re állítanánk?

A következő ábra néhány példát mutat a flex tulajdonság használatára. Figyeld meg, hogy a dobozok szélessége nem csak a flex értékétől, hanem a dobozok számától is függ!



Próbáld ki a flex működését oszlopos elrendezéssel is, majd állítsd vissza a soros elrendezést!

Megjegyzés: A flex tulajdonsághoz nem rendelhetünk negatív értéket.

## További flex beállítások

A flex tulajdonságnál két további értéket is megadhatunk.

A második érték (flex-shrink) azt adja meg, hogy ha kevés a hely, vagyis a dobozok szélességének összege nagyobb, mint a tároló, akkor milyen arányban legyenek keskenyebbek a dobozok. Ennek alapértelmezett értéke 1, vagyis mindegyik doboz szélességéből ugyanannyit vesz el, és ezt általában nem is kell módosítani. (Negatív érték itt sem adható meg.)

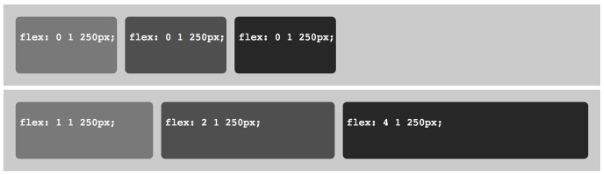
A harmadik érték (flex-basis) a doboz alap szélessége. A szélesség a tároló szélessége és a másik két érték szerint változhat.

Ha egy elemnél nem adjuk meg a flex tulajdonságot, akkor az alapértelmezett szélesség auto, vagyis a doboz tartalmához igazodik. Ha viszont megadunk flex tulajdonságot, de nem adjuk meg benne a harmadik értéket (flex-basis), akkor a böngésző erre 0-t feltételez.

Ha tehát nem állítunk be flex értéket egy elemre, akkor az ezt jelenti:  
flex: 0 1 auto;

Ha ezt írjuk egy flex elemre: flex: 2;  
akkor az ezt jelenti: flex: 2 1 0;

Nézzünk két példát a flex értékek használatára:

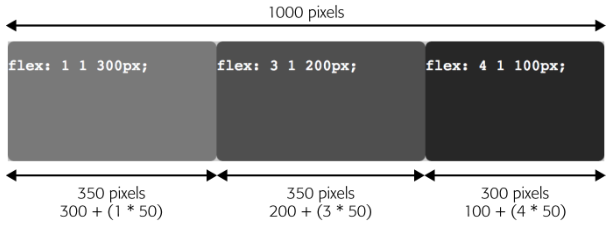


Az első sorban a dobozok szélessége nem nő (az első érték 0), az alap szélességek érvényesek. Ha a tároló szélessége 750px alá csökkenne, akkor ugyanannyit venne el minden doboz szélességéből.

A második sorban 1, 2, és 4 arányban nőtt meg a dobozok szélessége, hogy kitöltsék az egész sort. Ha a tároló szélessége 750px alá csökkenne, akkor itt is ugyanannyit venne el minden doboz szélességéből.

## Egy kis matematika

Nézzük meg, hogy pontosan hogyan határozza meg a böngésző az egyes dobozok szélességét!



A dobozok alap szélességének összege: 300+200+100=600px

A tárolóban rendelkezésre álló szélesség: 1000px

Mivel minden doboznál pozitív flex-grow érték van megadva, a dobozok kitöltik a rendelkezésre álló szabad helyet. A szabad hely mérete: 1000-600=400px

A szabad helyet a dobozok között 1, 3, 4 arányban kell elosztani. 1+3+4=8, ezért az első doboz a 400px 1/8 részét kapja (50px), a második 3/8-ot (150px), a harmadik pedig 4/8-ot (200px).

## Példa

Nézzünk egy egyszerű példát, hogyan lehet elkészíteni egy két oszlopos reszponzív elrendezést flexbox módszerrel!

1. Nyisd meg a pelda1 mappát a Visual Studio Code-ban és a böngészőben!
2. Próbáld ki különböző ablakszélességekkel!
3. Nézd át a HTML és a CSS kódot!
4. A main rész az ablak szélességének hány százalékát foglalja el (500px felett)?
5. Módosítsd az oldalt úgy, hogy az oldalsáv csak az ablak negyedét foglalja el!
6. Helyezd át az oldalsávot a bal oldalra! Figyelj a margókra! Keskeny ablaknál továbbra is a main rész után következzen az oldalsáv!
7. Mik azok, amit flexbox nélkül (float használatával) nehéz lenne megoldani?

## Önálló feladatok

Töltsd fel egy online tárolóhelyre ezt a feladatsort és az órán készített fájlokat! Otthon is nézd át őket!