Flexbox (rugalmas doboz)

Biztos vagyok benne, hogy nagyon sokan hallottatok már a CSS Flexboxról. Ennek az lehet az oka, hogy egy ismert, és nagyon praktikus eleme a CSS-nek: segítségével nagyon könnyedén lehet pozicionálni a képernyőnkön a különféle elemeket. Nem mindig praktikus a használata, lehetséges, hogy bizonyos esetben nem ez a legjobb megoldás, viszont kezdőként az emberben sokszor felmerül egy adott probléma esetén a kérdés: ezt a konkrét, specifikus problémát hogyan lehetne picit „elegánsabban” megoldani?

Mindenki, aki először találkozott a HTML oldalon található elemek pozícionálásának feladatával, biztosan emlékszik arra, hogy mennyire nehezen lehetett megérteni a különböző pozicionáló módszereket, azok egymásra hatását. A probléma gyökere abban keresendő, hogy **egy HTML-oldal feldolgozásakor a böngésző alapértelmezetten sorra veszi az egyes sor- és blokkszintű elemeket, és vagy egymás mellé helyezi el azokat (ezek az *inline elemek*), vagy egymás alá (ezek a *blokkszintű* elemek).**

A fenti problémát persze ***részben meg lehet kerülni olyan keretrendszerek használatával, mint a Bootstrap***, ahol kész CSS-osztályok állnak a rendelkezésünkre – ezek használata már sokkal rugalmasabb elrendezést tesz lehetővé. Természetes módon született meg az igény arra, hogy ez a rugalmasság már a CSS szintjén megjelenjen, ezért került bele a CSS3-ba a Flexbox elrendezés.

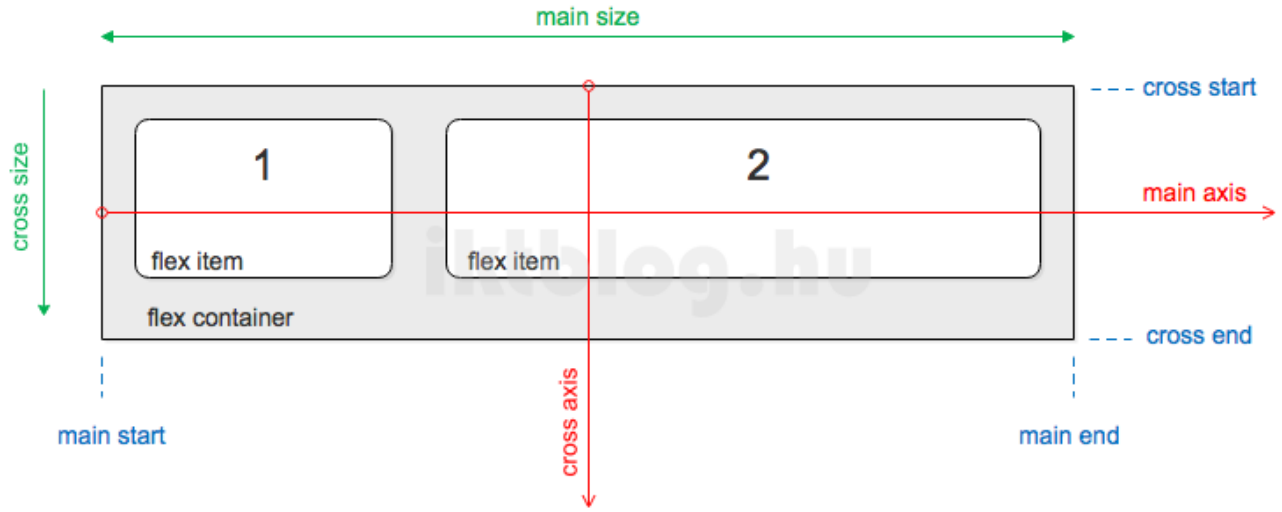
A Flexbox elnevezés a rugalmas dobozmodell (elastic box modell) rugalmas doboz moduljának (flexible box modul) rövidítése. Segítségével a HTML-elemek elhelyezkedését, rugalmas méretét és a közöttük lévő távolságokat könnyen állíthatjuk be.

A CSS flexbox tehát egy modern lehetőség arra, hogy az elemeink elrendezését módosítsuk vele a képernyőnkön. (Tulajdonképpen egy display érték, hasonlóan az inline-hoz, vagy a block-hoz.) Ennek a segítségével válik lehetségessé, hogy az elemeinket egy adott sorba, oszlopba rendezzük. Meghatározhatjuk a fő irányokat, az elemek térközeit, illetve azt is, hogy az elemek új sorba csússzanak-e, amennyiben már nem férnének ki, vagy pedig az elemek mérete csökkenjen, de azok maradjanak egy sorban.

A modell két alapeleme a **konténer** (***container***) és az **elem** (***item***). Értelemszerűen a konténer tartalmazza az elemeket, és képes azok szélességét és magasságát, sőt a sorrendjét is úgy megváltoztatni, hogy azok a lehető legjobban töltsék ki a rendelkezésre álló helyet. Képes akár az elemek szélességének növelésére vagy akár csökkentésére, az esetleges átfedések elkerülése érdekében.

Fontos tudni, hogy a flexbox elrendezés irányfüggetlen! (Amíg a blokkok a vertikális, addig az inline elemek horizontális elrendezésen alapultak.)

A flexbox működési elvét bemutató ábra



A flexbox használatához az alábbi ***fogalmak*** ismeretére lesz szükségünk:

- **main axis** (főtengely): a konténer elsődleges tengelye, amely mentén az elemek elrendeződnek (csak az ábrán van vízszintesen elhelyezve, mert értékét a **`direction`** tulajdonság segítségével függőlegesre is állíthatjuk)
- **main-start**, **main-end**: a konténer kezdetét és végét jelzi, ezen belül helyezkednek el a konténer elemei
- **main-size**: meghatározza a konténer méretét a főtengely mentén (ez a főtengely irányától függően lehet a szélessége vagy a magassága)
- **cross-axis** (keresztirányú tengely): a főtengelyre merőleges tengely, amelynek iránya a főtengely irányától függ
- **cross-start**, **cross-end**: mivel egy konténer elemekkel van megtöltve, ezért ezek az elem a sortörés miatt akár több sorban vagy több oszlopban is elhelyezkedhetnek – keresztirányban (azaz keresztirányú tengely mentén) az elemek kezdeti és végpontját jelöli
- **cross-size**: meghatározza a konténer méretét a keresztirányú tengely mentén (a főtengely irányától függően lehet a magassága vagy a szélessége)

Konténerek és elemek

A CSS Flexboxot úgy kell létrehozni, hogy egy szülőelemnek (vagyis a külső elemnek) a `display: flex` tulajdonságot adjuk. Ekkor minden egyes elem, ami közvetlenül ezen belül található, ennek megfelelően fog rendeződni.

HTML:

```
<div class="flex-container">
   <div>Elem 1</div>
   <div>Elem 2</div>
   <div>Elem 3</div>
</div>
```

CSS:

```
.flex-container {
   display: flex;
}
```

Mivel a flexbox főtengelyének alapértelmezett iránya a vízszintes, ***a konténer elemei is így rendeződnek el és szélességi értéküket a böngésző a beállított értékek*** (padding, margin, font-size, …) ***alapján számolja ki***. Az is látható, hogy az elemek nem töltik ki automatikusan a rendelkezésre álló területet, a konténer viszont igen (1. ábra). Ha viszont erre nincs szükség, akkor a ***flex-inline*** beállítást kell használni (2. ábra):

1. ábra: ***display: flex***

2. ábra: ***display: inline-flex***

(Általában egyébként az első változatot szoktuk használni.) Az elemek főtengely menti méretét a dobozmodell alapján számolja ki. Ezt felülbírálhatjuk az elemekre alkalmazott **`flex-basis`** tulajdonsággal:

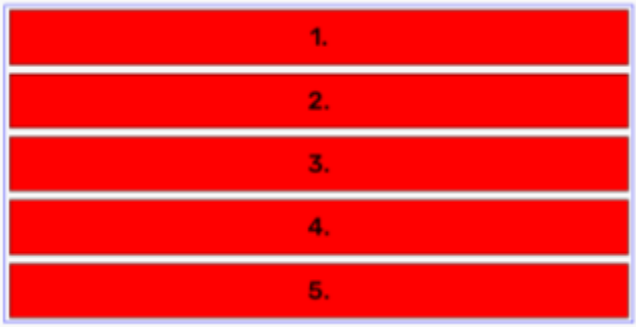
```
.belso_elemek { flex-basis: 150px; }
```

Ennek alapértéke az **`auto`**, ilyenkor a böngésző maga számolja ki, de megadhatunk konkrét méretet is.

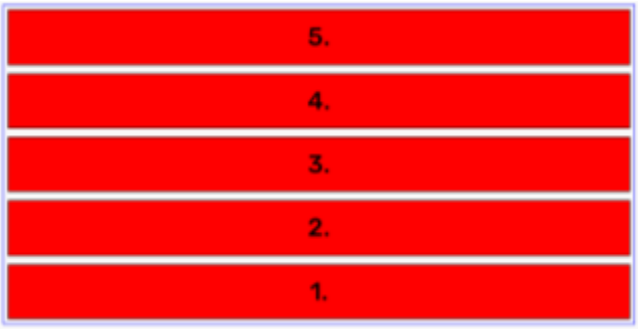
Azt már láttuk, hogyan történik a méretezés a főtengely mentén, de mi van a keresztirányban? ***Ha megnöveljük a konténer függőleges méretét, akkor azt tapasztaljuk, hogy a keresztirányú tengely mentén az elemek méretét hozzáilleszti a konténer méretéhez.*** Természetesen, ha valamelyik elemre konkrét méretet (vagyis most magasságot) állítunk be, akkor azon az átméretezés nem történik meg.

3. ábra: ***flex-direction: row***

4. ábra: ***flex-direction: row-reverse***



5. ábra: ***flex-direction: column***



6. ábra: ***flex-direction: column-reverse***

A tengelyek iránya

A bevezetőben láttuk, hogy két tengelyünk van: a főtengely (main-axis) és a rá merőleges, keresztirányú tengely (cross-axis). Alapértelmezetten a főtengely vízszintes, ezért a keresztirányú függőleges.

A főtengely irányát a szülőelemre beállított `flex-direction` tulajdonság segítségével tudjuk szabályozni.

Négyféle értéke lehet, és mindegyik elnevezés önmagáért beszél: row, row-reverse, column, column-reverse.

Az egyes flex-direction értékek jelentése:

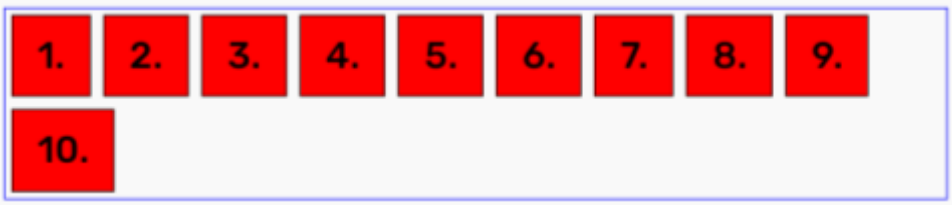
- **row** (alapértelmezett): a konténerben az elemek sorban, balról jobbra jelennek meg (3. ábra)
- **row-reverse**: a konténerben az elemek sorban, jobbról balra, vagyis fordított sorrendben jelennek meg (4. ábra)
- **column**: a konténerben az elemek felülről lefelé jelennek meg (5. ábra)
- **column-reverse**: a konténer elemei alulról felfelé jelennek meg (6. ábra)

Az elemek sortörése, elemek sorrendje

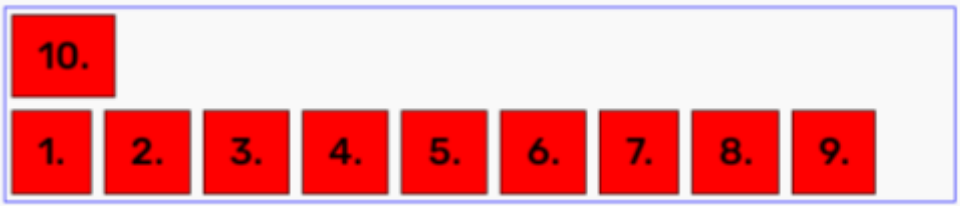
Mi történik, ha a főtengely mentén nem férnek el az elemek az oldalon? Méretezzük át az oldalt úgy, hogy a fenti példákban a konténer elemei vízszintesen (vagyis alaphelyzetben a főtengely mentén) nem férnek el! Figyeljük meg, hogy mi történik a kilógó elemekkel! Azt tapasztaljuk, hogy jelen esetben elég csúnyán kilógnak az oldalból. Ennek megoldására használjuk a `flex-wrap` tulajdonságot:

- **nowrap** (alapértelmezett érték): nem történik sortörés (7. ábra)
- **wrap**: a kilógó elemeket új sorba tördeli (8. ábra)
- **wrap-reverse**: a kilógó elemeket új sorba helyezi, de fordított sorrendben (9. ábra)

7. ábra: ***flex-wrap: nowrap***



8. ábra: ***flex-wrap: wrap***



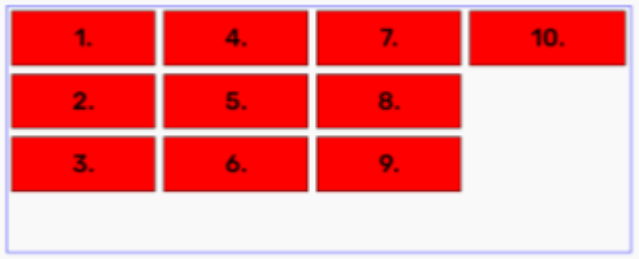
9. ábra: ***flex-wrap: wrap-reverse***

A főtengely irányától függően a sortörés függőleges irányban is történhet!

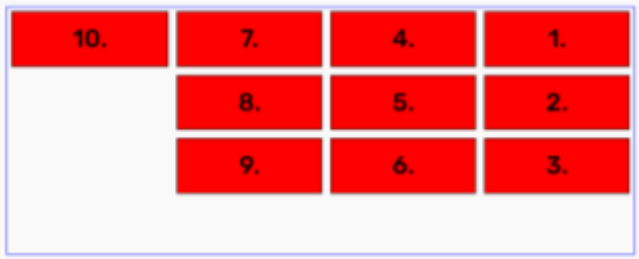
Lehetőségünk van a főtengely irányát és a sortörést egyetlen tulajdonságon keresztül is beállítani, erre szolgál a `flex-flow` tulajdonság. ***Ennek a tulajdonság két értéket kell megadni: az első a főtengely irányát, a második pedig a sortörés módját jelenti.*** (Az értékek megegyeznek az egyedi beállításokkal.)

Ha például a főtengely függőleges és az aszerinti sortörést szeretnénk megadni, akkor az alábbi esetek jöhetnek szóba:

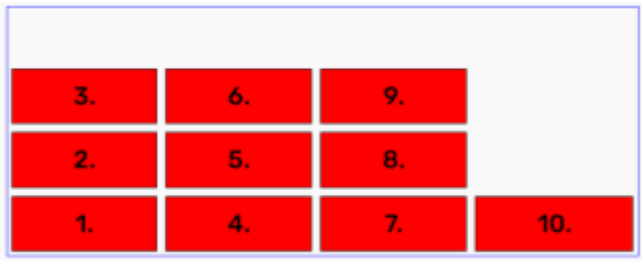
- **column nowrap** = az elemeket egymás alá helyezi (az egyes elemek szélessége megegyezik a szülőelem teljes szélességével) és előfordulhat, hogy a szülőelem magassága nem elegendő az egymás alá helyezett elemeknek, ezért az utolsó néhány elem akár ki is csúszhat a szülőelemből
- **column wrap** = az elemeket egymás alá helyezi a sorrendjük szerint, hogy ha valamelyik már nem fér el az oszlopba, akkor azt a következő oszlopba rendezi (10. ábra)
- **column wrap-reverse** = az elemeket egymás alá helyezi a fordított sorrendjükben úgy, hogy az első elemet a szülődoboz jobb szélén kezdi és ha a jobb oldali oszlop betelt, akkor a következő oszlopot attól balra helyezi el (11. ábra)
- **column-reverse wrap** = az elemeket egymás alatt fordított sorrendben helyezi el mindaddig, amíg az adott oszlopba kiférnek, majd jobbra indulva tördeli meg azokat (12. ábra)
- **column-reverse wrap-reverse** = az elemeket egymás alatt fordított sorrendben helyezi el és az első oszlopot a jobb szélre rendezi (13. ábra)



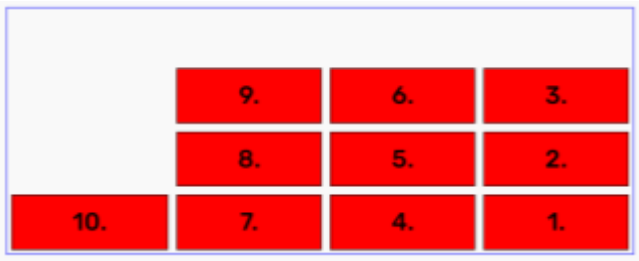
10. ábra: ***flex-flow: column wrap***



11. ábra:
flex-flow: column wrap-reverse



12. ábra:
flex-flow: column-reverse wrap



13. ábra:
flex-flow: column-reverse wrap-reverse

A konténeren belüli elemek sorrendjének meghatározásához egy másik lehetőség is a rendelkezésünkre áll. Az elemeket sorszámmal azonosított csoportokra sorolhatjuk az `order` tulajdonsággal (a szám negatív, nulla vagy pozitív egész érték lehet). Azok az elemek, amelyek kisebb számú csoportokba vannak sorolva, a megjelenítésben előrébb kerülnek, mint a nagyobb számúak. Az azonos csoportba tartozó elemek sorrendjét természetesen az határozza meg, ahogyan azok egymás után jönnek. (Persze ekkor is ügyelni kell arra, hogy az alapsorrendet a tengelyek iránya és a sortörés iránya határozza meg először, a csoportokba tartozást csak ezek után értékeli ki a böngésző.) Alapértelmezés szerint miden elem a 0-s csoportba tartozik.

Az elemek igazítása

Annak érdekében, hogy az elemek igazítását megértsük, érdemes levenni az elemekre beállított margó értékét. ***Margó nélkül az elemek alapértelmezés szerint szorosan egymás mellett, balról jobbra helyezkednek el a vízszintes főtengely mentén.*** Most ezekre végezzük el az igazítás beállítását, amelyhez négyféle tulajdonság is a rendelkezésünkre áll:

- **`justify-content`**: az összes elemet együtt kezelve a konténer teljes szélességében igazítja, a második három az elemek közötti térközt szabályozza (14-19. ábra)
  - **flex-start**: mintha balra lenne az összes elem csúsztatva
  - **center**: mintha az összes elem együtt egyetlen blokkot alkotva középre lenne rendezve
  - **flex-end**: mintha jobbra lenne az összes elem csúsztatva
  - **space-around**: minden elem között azonos távolság van, de az első elem előtt és az utolsó elem után csak fele akkora (vagyis minden elem bal és jobb oldalára is ugyanakkora margót helyez el)
  - **space-between**: az első és az utolsó elem a konténer széleihez illeszkedik, de a többi elem között azonos a távolság
  - **space-evenly**: mindenütt azonos az elemek közötti távolság, így az első elem előtt és az utolsó elem mögött is

14. ábra: ***justify-content: flex-start***

15. ábra: ***justify-content: center***

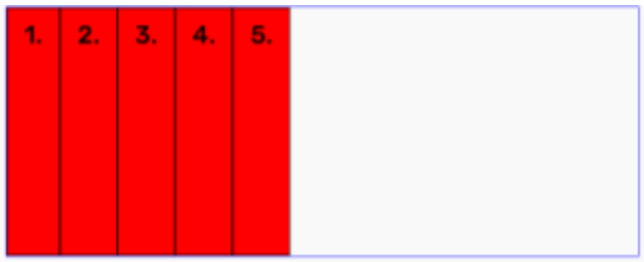
16. ábra: ***justify-content: flex-end***

17. ábra: ***justify-content: space-around***

18. ábra: ***justify-content: space-between***

19. ábra: ***justify-content: space-evenly***

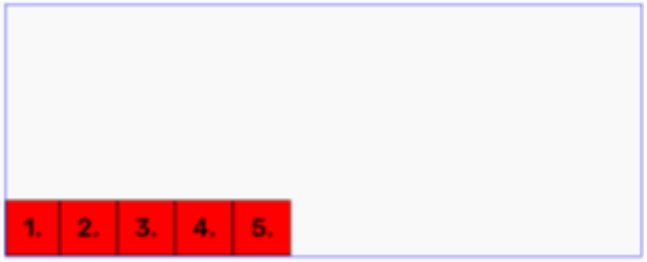
- **`align-items`**: az összes elemet a keresztirányú tengely mentén (vagyis a főtengelyre merőlegesen) igazítja (20-23. ábra)
  - **stretch** (alapértelmezett érték): a keresztirányú tengely mentén megnyújtja az elemeket a szülődoboz magassági méretével azonos értékben
  - **flex-start**: a keresztirányú tengely szerinti felülre igazítást végez (alaphelyzetben tehát a konténer tetejéhez igazítja az elemeket)
  - **center**: a keresztirányú tengely szerinti középre igazítást végez (alaphelyzetben tehát a konténer függőleges közepére igazítja az elemeket)
  - **flex-end**: a keresztirányú tengely szerinti alulra igazítást végez (alaphelyzetben tehát a konténer aljához igazítja az elemeket)



20. ábra: ***align-items: stretch***

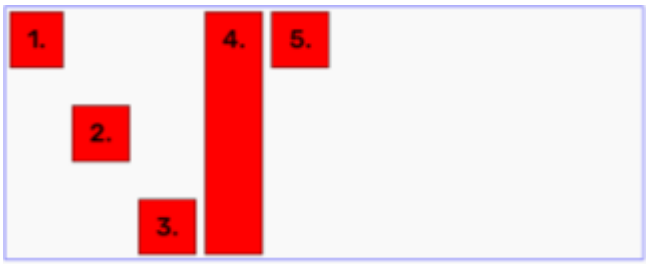
21. ábra: ***align-items: flex-start***

22. ábra: ***align-items: center***



23. ábra: ***align-items: flex-end***

- **`align-self`**: az egyes elemeket igazítja a tengelyek metszéspontjához képest (azaz a keresztirányú tengely mentén), tehát segítségével az egyes elemeknek külön is megadhatjuk az igazítását; szemléltetésükhöz érdemes az egyes elemek magasságát is megváltoztatni (24. ábra)
  - **flex-start**: az elem a konténer tetejéhez igazodik
  - **center**: az elem a konténer függőleges közepéhez igazodik
  - **flex-end**: az elem a konténer aljához igazodik
  - **stretch**: az elem a szülőelem magasságával egyező nagyságúra nyújtott
  - **baseline**: alapvonalhoz igazított



24. ábra: az ***align-self*** különféle beállításai (sorrendben: flex-start, center, flex-end, stretch és baseline)

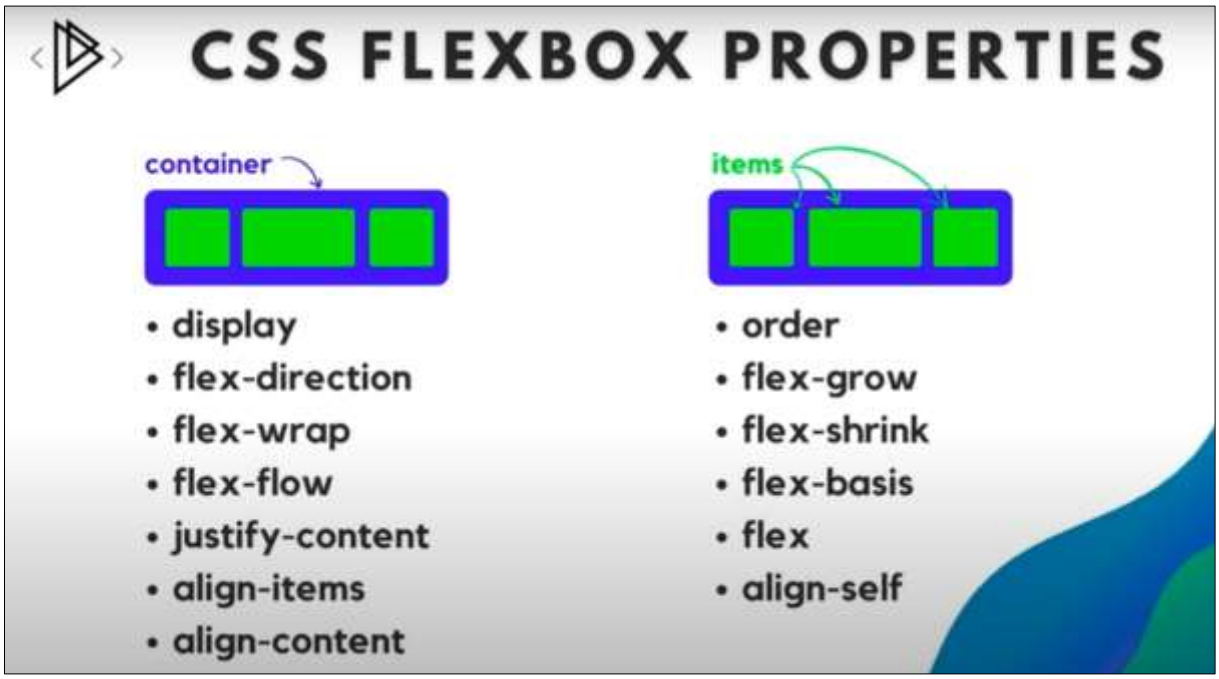
- **`align-content`**: az elemek közötti szabad helyeket határozza meg; segítségével azt befolyásolhatjuk, hogy ha sortörés miatt több sorban jelennek meg az elemek, a sorok milyen messze legyenek egymástól
  - **flex-start**: az oszlop tetejéhez rendezi az oszlopelemeket
  - **center**: az oszlop függőleges közepéhez rendezi az oszlopelemeket
  - **flex-end**: az oszlop aljához rendezi az oszlopelemeket
  - **stretch**: az oszlopban lévő egyes elemek magasságát úgy változtatja meg, hogy azok kitöltsék a konténer teljes magasságát
  - **space-between**: az adott oszlop első eleme az oszlop tetejéhez, az utolsó eleme az oszlop aljához igazodik, a többi aközött pedig azonos a távolság
  - **space-around**: az elemek felett és alatt azonos térközt helyez el

Az elemek rugalmas méretezése

A korábbi példákat áttekintve azt láthatjuk, hogy a konténeren belül az egyes elemek elhelyezkedése és az elemek közötti térköz számos módon változtatható. Elvárható így az is, hogy az elemek méretét is hasonló rugalmassággal szabályozhassuk. Korábban már láttuk a **`flex-basis`** tulajdonságot, amellyel a konténeren belül az elemek kezdőméretét tudtuk beállítani. Ez a méret azonban rugalmasan változtatható a **`flex-grow`** és a **`flex-shrink`** jellemzőkkel.

- A **`flex-grow`** tulajdonság azt határozza meg, hogy az egyes elemek mekkora részt kapjanak a konténer maradék térközéből. (A maradék térköz az, amit a kezdeti méretükkel nem töltenek ki.)
  - ha ez mindegyik elemnél 1: a maradék térköz egyenletesen lesz elosztva
  - ha valamelyik elemnél 2: az elemnek kétszer akkor térköz jut, mint a többiek számára
  - értéke nemcsak egész, hanem törtszám is lehet
- A **`flex-shrink`** tulajdonság akkor kap szerepet, amikor az elemek nem férnek el a konténer főtengelye mentén és nincs bekapcsolva a wrap. Ha definiáljuk ezt a tulajdonságot az elemeken, akkor a böngésző megnézi, hogy menynyivel nyúlnak túl az elemek a konténeren és mindegyikük méretét az e tulajdonságban megadott értéknek megfelelő arányban csökkenti. (Vagyis minél nagyobb ez az érték, annál nagyobb arányban kerül csökkentésre az elem.)

Összefoglaló



Segítségek a flexbox tanulásához:

A flexbox tulajdonságainak megtanulásához jó segítséget jelent a Flexbox Froggy (https://flexboxfroggy.com/#hu) oldal.

videó: Flexbox CSS In 20 Minutes (https://www.youtube.com/watch?v=JJSoEo8JSnc)