09 - CSS

Előadás demó leírása a Mobil- és webes szoftverek c. tárgyhoz

Gincsai Gábor

2017.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

1117 Budapest, Magyar tudósok körútja 2.
Q. ép. B. szárny 207. • www.aut.bme.hu
Telefon: 463-2870 • Fax: 463-2871

Bevezetés

Ebben a dokumentumban röviden áttekintjük az előadáson bemutatott demókat, hogy az a későbbiekben otthon önállóan is el lehessen végezni.

Célkitűzés

A demók elvégzésének segítségével gyakorlatiasan meg lehet ismerkedni a CSS készítésének folyamatával, ki lehet próbálni az előadáson elhangzott száraznak tűnő fóliákat.

A demók kifejezetten olyan hallgatóknak szólnak, akiknek a CSS még újdonság, abban tapasztalatuk csekély.

Miért érdemes megismerkedni a CSS-sel?

A webes fejlesztés során, legyen az kliens vagy szerver oldali kód, minden esetben a fejlesztőkre hátul a megtervezett design megvalósítása. Ahhoz, hogy ezt karbantartható módon szépen és érthetően tudjuk megoldani, tisztában kell lenni a leggyakoribb problémákkal és azok megoldásával.

A demókban olyan eseteket tekintünk át, melyek gyakran előfordulnak és bár megoldásuk nem bonyolult találgatással sok idő alatt lehet csak rájönni.

01 – Block elem magassága

Ha van egy blokk elemünk, pl div és arra nem fix pixelben, hanem százalékosan adjuk meg a magasságot, akkor az nem állítódik be.

A demó kód, ahol láthatjuk, hogy nem állítódik be a magasság: https://codepen.io/gincsai/pen/rGqbJQ

A probléma oka, hogy a szülő magassága „auto”-ra van téve, hiszen nem adtuk meg, és e miatt a szülő magassága annyi lesz, hogy a benne lévő elemeknek az éppen elegendő legyen.

Tehát a megoldás, hogy a fában végig (vagy legalább addig ahol pixelben van megadva a magasság) az összes szülő elemeknek meg kell adni százalékosan a magasságot.

A helyesen működő kód: https://codepen.io/gincsai/pen/PJxBGx

02 – Margin collapse

Ha két egymás követő blokk elemre alsó és felső margint adunk meg, akkor a két elem között a margó összeolvad és a két elem közötti távolság a két margó közüli nagyobbra fog beállítódni.

Példa a margók összeolvadására: https://codepen.io/gincsai/pen/OxawBK

Azonban ha a fenti kódot módosítjuk úgy, hogy a .box osztályon legyen egy border is, akkor már nem olvadnak össze a margók, hiszen közbe meg kell jeleníteni egy bordert is.

03 – Padding

Ha készítünk egy oldalt, ahol float segítségével egy-egy 50% széles div-et egymás mellé helyezünk, akkor azokat szépek ki fogják tölteni a rendelkezésre álló helyet.

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

Azonban, ha a demoban módosítjuk a kódot úgy, hogy az 50% szélesség mellé még 2px paddingot is beállítunk, akkor azt láthatjuk, hogy a két div egymás alá csúszik.

Ennek az oka a box méretezésben van, ami alapértelmezés szerint content-box, tehát nem számolja bele a paddingot. Megoldás lehet:

- ha a szélességet a width: calc(50 % - 4px) segítségével számoljuk
- vagy átállítjuk, hogy a boksz-sizing: border-box legyen

04 – Menu

Az inline-block elemek segítségével nagyon szép menüt lehet készíteni, a nélkül, hogy float-okat kellene kezelni és a függőleges igazítást is meg lehet egyszerűen oldani a vertical-align segítségével.

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

Azonban ha megnyitjuk a fenti linket azt tapasztaljuk, hogy a menü elemek között egy fehér rész található, holott sehol sem állítottunk margint.

A probléma, hogy inline-block esetén a szóközöket nem dobja ki a böngésző, hanem csak egy elemmé sűríti össze.

A szóközök / white space-ek pedig az egyes sortörések az li-k között.

Tehát a lehetséges megoldások:

- Kiszedjük az összes white space-t és egy sorban lesz a teljes ul-ünk. (Nem szép, de működik)
    - https://codepen.io/gincsai/pen/GMwwWY
- Kikommentezzük a sortöréseket. (Továbbra sem szép)
    - https://codepen.io/gincsai/pen/BwGGRb
- Vagy az ul elemen a betű méretet 0-ra vesszük, majd az li-nél visszaállítjuk az eredetire.
    - https://codepen.io/gincsai/pen/zEMMLd
    - Problémák a megoldással:
        - tudunk kell, hogy mennyi volt az eredeti betűméret a nullázás előtt.
        - Nem használhatjuk az em-en, mert a szülő ugye 0, és 0 * valami az 0.
        - Új böngészőben a megoldás a „rem” ami a body betűméretéhez képest relatív.

05 – Reset CSS

A reset CSS-ek lényege, hogy a böngésző alap beállításokat kiüssék. (Mivel kiütni ténylegesen nem lehet, ezért valójában felülírja üres értékkel mindet.)

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

A fenti kódban látható, hogy a h1 elem is normál betűméretet használ, a vastag betű sem vastag, de a linkek továbbra is alá vannak húzva.

06 – Normalize CSS

A normalize CSS lényege, hogy egységesíti a böngészők alapértelmezett stíluslapján. Tehát minden értéket felüldefiniál, amit fejlesztés közben majd be tudunk állítani a nekünk megfelelő értékekre.

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

Pl.: A line-height helyesen számítódik akkor is ha van alsó vagy felső index.

07 – Blokk folyam és a függőleges igazítás

A blokk elemek egymás alatt helyezkednek el, így a box-modell tulajdonságok változása szinte mindig befolyásolja az utánuk jövő elemek elhelyezkedését.

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

Változtassuk a .box-special tulajdonság értékét és látható lesz, hogy a legalsó téglalap követi a lila téglalap magasságát és ettől függően kilóg a fehér részből vagy nem.

08 – DISPLAY: INLINE HASZNÁLATA

Például a navigációs linkeket nagyon egyszerűen meg lehet oldani a display: inline segítségével.

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

09 – DISPLAY: BLOCK HASZNÁLATA

Ha az szeretnénk, hogy az egymás után következő elemek új sorba törjenek, akkor ne a
-t használjuk, hanem egy block elembe tegyük a tartalmat.

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

10 – POSITION: RELATIVE

A position: relative elemek egy új rétegre kerülnek és az eredetileg neki szánt hely üresen marad.

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

Ilyen esetben ha nem adunk meg z-index-et akkor is a többi elem fölött lesz a kép, hiszen egy új rétegre kerül.

A példában azért csúszik ki a kép a rózsaszín dobozból, mert az img elemre meg van adva, hogy bal oldalról 50px távolságot tartson. Ha azt 0-ra állítjuk nem fog kilógni.

Demo2: https://codepen.io/gincsai/pen/QqzOzw

A fenti demoban két képünk van, ahol az első (egy téves pozícionálás miatt), kitakarja a második képet, pedig a második képen a z-index: 9999 meg van adva. A gond az, hogy ilyen esetben a z-index nem segít, a második képet is relative kell pozícionálni, ha azt szeretnénk, hogy az legyen felül és a z-index nem is szükséges.

11 – POSITION: ABSOLUTE

Demo: https://codepen.io/gincsai/pen/eGbyVV

A fenti demoban az látható, hogy a position: absolute megadása esetén nem a bal felső sarokba kerül automatikusan a tartalom. Nekünk kell kézzel megadni, hogy top: 0 és akkor kerül egy magasságba a szülő elemmel.

Na de melyik szülő elemmel? Az absolute pozíció mindig ahhoz a legközelebbi szülőhöz képest abszolút, amin position: relative van állítva. Próbáljuk is ki, hogy a figure-ra megadjuk, hogy postion: relative és akkor a kép teteje a rózsaszín téglalappal lesz egy vonalban.

Abszolút pozícionálás esetén megadhatjuk egyszerre a top, bottom, left, right értékeket is. Ebben az esetben a teljes szélességet és magasságot ki tudjuk tölteni.

Demo: https://codepen.io/gincsai/pen/jGXYdL

Abban az esetben ha a képeknek nem szeretnénk megadni a szélességét, akkor az inline-block lesz a helyes irány, mert akkor tartalom szerinti szélességet kapunk, és bármilyen széles / magas képet teszünk be, akkor is működni fog a kód.

Demo: https://codepen.io/gincsai/pen/EwGoee

Figyeljük meg, hogy csak annyi változott, hogy a figure inline-block lett és nincs megadva a szélesség.

12 – POSITION: FIXED

Akkor jó, ha olyan kódot szeretnénk készíteni, hogy például scrollozáskor a fejléc ott maradjon fixen a helyén.

Demo: https://codepen.io/gincsai/pen/OxrQMp

13: Z-INDEX

Alapértelmezés szerint az elemek úgy kerülnek egymás fölé, amilyen sorrendben megadtuk őket a HTML-ben.

Demo: https://codepen.io/gincsai/pen/JrwpEP

Nem kell nagyon nagy z-indexeket használni, a -1, 0, 1-el már befolyásolható a megjelenítési sorrend. Ennek kipróbálásához kommentezzük ki a z-index-et tartalmazó CSS szabályokat.

Ezen felül az is fontos, hogy a z-index számításakor a szülők z-indexe is számít. Ezt láthatjuk ebben a demóban: https://codepen.io/gincsai/pen/LzMQLV

Figyeljük meg, hogy a .container z-indexe 3, e miatt a section, aminek a z-indexe 5 lesz legfelül, függetlenül attól, hogy a .blue osztályban 6-os z-indexet adtunk meg, mert a blue div, a container-en belül van ami miatt 3-as a z-indexe és a 6 csak a container-en belüli sorrendet befolyásolja.

14 – FLOAT ALAPÚ LAYOUT

Az alábbi példában azt látjuk, hogy két div float-tal egymás mellé van rendezbe. (Egyik balra, másik jobbra.)

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

Azonban amint margin-t állítunk a div-ekre (pl.: jobb és bal oldalra 2,5%-ot), akkor egymás alatt lesznek a div-ek, mert nem férnek ki egymás mellé. Ekkor le kell csökkenteni a div-ek szélességét (45%-ra), hogy elférjelek egymás mellett.

Azonban ha levesszük a szélességet, azt fogjuk látni, hogy középen dupla akkora a távolság a két elem között.

Ilyen esetben ha azt szeretnénk, hogy mindenhol ugyanannyi legyen a távolság, a megoldás, hogy mind a két div-et balra floatoljuk, és csak bal oldali margót állítunk nekik.

Ez a példa a @media tag használatát is mutatja, ami a reszponzív layout alapja. Lényege, hogy megadható, hogy ha a szélesség maximum 800px, akkor milyen szélességeket állítson be. A mobil layoutot ezzel lehet megvalósítani. (A bootstrap is ezt használja.)

Ha kikommentezzük a @media részt a példában, és 800px-nél kisebbre vesszük az oldalt, akkor látható is, hogy automatikusan átrendeződik a layout.

Több hasábos float alapú layoutra pedig ezt a példát érdemes megnézni: https://codepen.io/gincsai/pen/yzGxxz

A float alapú layoutnak van egy olyan problémája, hogy ha a float-olt elemek nem töltik ki teljesen a rendelkezésre álló szélességet, akkor a többi eleme felcsúszik közéjük. Erre egy példa: https://codepen.io/gincsai/pen/zEyLjN

Ilyen esetben a megoldás lényege, hogy a float-okat törölni kell az előtt az elem előtt, amit már mindenképpen új sorba szeretnénk tenni, jelen esetben a footer-en.

Egy másik gyakori esetet, hogy a float-olt elemek szülő elemének a magassága 0 lesz. Ehhez a következő példát nézzük meg: https://codepen.io/gincsai/pen/JrwBBy

Itt ha beállítjuk a .container-re a bordert máris látható a probléma, amit többféleképpen is meg lehet oldani.

- Beteszünk a HTML-be egy üres div-et, amin a clear: both tulajdonság be van állítva.
- Ha a HTML-hez nem szeretnénk hozzányúlni, akkor pedig a CSS-es pszeudo szelektorokkal is létre lehet hozni egy HTML taget, amire megadjuk CSS-ből a clear: both-ot. Ezt a .container: after résszel tudjuk megtenni.
- Beállítjuk a container-re az overflow: hidden tulajdonságot*
    - Ez a megoldás azért működik, mert az overflow: hidden egy új block formatting contextet hoz létre.

15 – INLINE-BLOCK ALAPÚ LAYOUT

A https://codepen.io/gincsai/pen/oGJMeG példában azt látjuk hogy a fenti float-olt layoutot inline-block-kal is meg lehet csinálni. A kód mennyisége hasonló. Itt nem kell clearfix, működik a vertical-align viszont a betűmérettel gondok lehetnek. (Le kell venni 0-ra majd visszaállítani az eredetit.)

Azonban tipikus felhasználási területe ennek a layoutnak, ha többoszlopos elrendezést szeretnénk úgy készíteni, ahol az elemek magassága nem azonos, mint ebben a példában: https://codepen.io/gincsai/pen/WZLKxP

Ha a .gallery-item-re beállítanánk a float: left-et láthatóvá válik, hogy a float-olás tönkreteszi a layoutot.

16 – TABLE ALAPÚ LAYOUT

A talbe alapú layout tipikus felhasználása: https://codepen.io/gincsai/pen/zEyMyZ

Fontos, hogy a fenti példában lássuk, hogy nem a html <table> <tr> <td> elemekkel hoztuk létre a táblázatot, hanem CSS-ből a display tulajdonság segítségével.

Ennek a megoldásnak az előnye, hogy a függőleges igazítás működik benne: https://codepen.io/gincsai/pen/PJXxLL

Tudunk vele egyszerűen több oszlopos elrendezést is készíteni: https://codepen.io/gincsai/pen/VMqVOB ahol a cellák magassága (sormagasság) azonos lesz.

Akár egy navigációs fejlécet is készíthetünk vele: https://codepen.io/gincsai/pen/VMqVOB ahol a cellákon belül középre van vízszintesen igazítva maga a link.

De akár egy bonyolult reszponzív layoutot is készíthetünk vele: https://codepen.io/gincsai/pen/RLEEbE

- A fenti példa IE-ben kis méreten kicsit szétesik, ami jól szemlélteti, a böngészők közötti eltéréseket.