SZABVÁNYKÖVETŐ STATIKUS HONLAPOK SZERKESZTÉSE

HTML5 + CSS3 + SVG 2

1.0 Aurora

TARTALOM

TARTALOM ... 2
ELŐSZÓ ... 5
1. ALAPOK ... 7
1.1. Tartalom és forma: HTML és CSS ... 7
1.2. A weblapkészítés hardver és szoftver igénye ... 8
1.3. Statikus webhelyek kialakításának szokásos módja ... 15
2. HTML5 ... 16
2.1. A HTML nyelvtana ... 16
2.2. A weboldal szerkezete ... 18
2.3. Az első konkrét weblap ... 22
2.4. Alapvető HTML címkék ... 23
2.5. Listák ... 27
2.6. Táblázatok ... 33
2.7. Képek ... 48
2.8. Mozgóképek és hang ... 51
2.9. Beágyazott böngészőtartalom ... 56
2.10. Karakter entitások ... 58
2.11. Tartalmi/formázási HTML címkék ... 60
2.12. A weboldal egyes részeinek formázását elősegítő HTML címkék ... 68
2.13. Hivatkozások ... 76
2.13.1. Külső hivatkozások ... 76
2.13.2. Belső hivatkozások ... 77
2.13.3. Képek használata hivatkozásokhoz ... 82
2.13.4. Hivatkozások használata multimédiához és e-mail-hez ... 85
2.13.5. Hivatkozás használata letöltéshez ... 85
2.13.6. Képtérképek ... 86
2.14. Űrlapok ... 89
2.14.1. Az *input* címke ... 89
2.14.2. Egyéb címkék ... 97
2.14.3. Egy űrlap megszerkesztése ... 103
2.15. Meta-adatok ... 106
2.16. Webhely hierarchikus mappaszerkezettel ... 107
2.17. HTML5 összefoglaló ... 113
3. CSS3 ... 119
3.1. Kitöltőszöveg és kitöltőkép ... 119
3.2. Képformátumok ... 120
3.3. Színek ... 122
3.3.1. RGB(A) ... 122
3.3.2. HSL(A) ... 128
3.3.3. Színharmóniák ... 131
3.4. Értékek és mértékegységek ... 136
3.5. Az elemi doboz-modell ... 138
3.6. A CSS nyelvtana ... 140
3.7. A CSS hierarchiája ... 149
3.8. Betűtípusok ... 151
3.9. Színek definiálása ... 159
3.10. Háttérszín definiálása ... 160
3.11. Blokkszintű elemek elhelyezése I. ... 161
3.11.1. Méretek ... 162

3.11.2. Elhelyezés ... 163
3.12. Szöveg formázása ... 164
3.12.1. Szöveg pozícionálása és díszítése ... 165
3.12.2. Szöveg tördelése és elválasztása ... 170
3.12.3. Szöveg árnyéka ... 171
3.13. Listák formázása ... 175
3.14. Standard vonaltípusú szögletes szegélyek ... 178
3.15. Standard vonaltípusú lekerekített szegélyek ... 187
3.16. Képből készített szegélyek ... 193
3.17. Körvonal ... 200
3.18. Blokkszintű elemek elhelyezése II. ... 201
3.18.1. Elhelyezés módja ... 201
3.18.2. Láthatóság ... 211
3.18.3. Láthatósági sorrend ... 211
3.19. Doboz árnyékok ... 216
3.20. Háttérképek definiálása ... 219
3.20.1. Háttérképek ismétlődése ... 220
3.20.2. Hátterek hatóköre ... 224
3.20.3. Háttérképek helyzete ... 225
3.20.4. Háttérképek mérete ... 227
3.20.5. Háttérképek görgethetősége ... 228
3.20.6. Összevontan megadható háttérkép tulajdonságok ... 229
3.20.7. Több háttérkép egyidejű alkalmazása ... 230
3.20.8. Hátterek keverése ... 231
3.21. Táblázatok formázása ... 233
3.21.1. Nem táblázat-specifikus tulajdonságok alkalmazása ... 235
3.21.2. Táblázat-specifikus tulajdonságok alkalmazása ... 240
3.22. Túlnyúlás ... 244
3.23. Egyes elemekhez kapcsolódó kurzor-megjelenítés ... 248
3.24. Egérkurzorral egyes elemek megjelenítésének a megváltoztatása ... 249
3.24.1. Hirtelen átmenet ... 250
3.24.2. Folyamatos átmenet ... 252
3.24.3. Animáció ... 256
3.25. Többhasábos elrendezés ... 263
3.26. Színátmenetek ... 272
3.26.1. Lineáris színátmenetek ... 272
3.26.2. Sugárirányú színátmenetek ... 276
3.26.3. Periódikusan ismétlődő színátmenetek ... 279
3.27. Síkbeli transzformációk ... 285
3.28. Térbeli transzformációk ... 291
3.29. Speciális formázások pszeudo-elemekkel ... 300
3.30. CSS alapbeállítás ... 306
3.31. Űrlap formázása ... 309
3.32. Legördülő menük ... 311
3.33. Weblap elrendezések ... 313
4. SVG 2 ... 330
4.1. Alakzatok ... 330
4.2. Szöveg ... 343
4.3. Szöveg illesztése alakzatokhoz ... 350
5. FÜGGELÉK - Történeti összefoglaló ... 353

5.1. Előzmények I. (Internet) 353
5.2. Előzmények II. (hiperszöveg, jelölőnyelv) 354
5.3. Előzmények III. (HTTP, HTML, www) 354
5.4. Előzmények IV. (W3C, CSS, PNG, SVG) 355
5.5. Előzmények V. (XHTML) 356
5.6. Rövid böngészőtörténelem 358
5.7. A jelen és belátható jövő (HTML5, CSS3, SVG 2) 360

ELŐSZÓ

45 éve kezdett működni két amerikai kutatóhely számítógéprendszerének alapvetően katonai célokat szolgáló összekapcsolásával az Internet őse, az akkori szövetségi kormányzati védelmi ügynökség (Advanced Research Project Agency = ARPA) nevét viselő ARPANET (jelenleg DARPA az ügynökség elnevezése).

25 éve szünt meg az ARPANET, átadva helyét a nem-katonai alkalmazásokban egy amerikai tudományos alapítvány (National Science Foundation = NSF) nevét viselő NSFNET-nek és a kialakuló magáncéges gerinchálózatoknak. Így létrejött a tudományos és üzleti, kormányzati és magán, helyileg és globálisan összekapcsolt számítógéphálózatok nyílt, átfogó rendszere, mely a csatlakoztatott számítógépek közötti kapcsolat révén univerzális adatátvitelt nyújt.

Ugyanebben az évben javaslat készült a CERN-ben (Centre Européen pour la Recherche Nucléaire = CERN) - mely az akkori Internet egyik legnagyobb európai csomópontja volt - egy Internet-en alapuló információs rendszer kiépítésére, mely egy módosított jelölőnyelv (HTML) és erőforrás-azonosítók (URL) együttes használatával az intézet tudományos eredményeinek publikálására és az azokhoz való azonnali és közvetlen hozzáférésre irányult. Bár az első szerver csak két év múlva kezdett üzemelni – honlapjának címe *http://info.cern.ch/hypertext/WWW/TheProject.html* volt - a 25 évvel ezelőtti időpontot tekintik a világháló születésnapjának.

20 éve rendezték meg Genfben a CERN szervezésében az első nemzetközi világhálós konferenciát (First International WWW Conference).

Ugyanebben az évben jött létre egy-egy amerikai és európai központtal egy állandó nemzetközi non-profit intézmény, a World Wide Web Consortium (általánosan használt rövidítése „W3C”), melynek célja a világháló folyamatos fejlesztésének és szabványosításának elősegítése. A fennállásának 20. évét idén ünneplő szervezet koordinálja a jegyzetben tárgyalt HTML5, CSS3 és SVG 2 kialakítását, propagálja szabványos használatukat.

A HTML5 fejlesztésének kezdete 2004-re, a CSS3-é még régebbre, 1999 végére vezethető vissza, így 10 ill. 15 évükkel a leghosszabb ideig alakítgatott web-es szabványoknak tekinthetők.

Az egymással versengő böngészőgyártók szolgáltatásaikban a HTML5 és CSS3 kompatibilitást 2011-től kezdték hangsúlyosan kezelni, és a Microsoft csatlakozása ezen technikáknak (beleértve az SVG-t is) a termékeiben való alkalmazásához mérföldkövet jelent a web-szerkesztés szabványosítása útján. Minden jelentős mobileszköz támogatja a HTML5 technológiát, a CSS3 (többek között) segíti az asztali és mobil eszközökre való közel egységes tervezést, az SVG pedig biztosítja a jól átméretezhető grafikát, így a mobil platformok (táblagép és okostelefon) gyors terjedése tovább növeli ezen technikák alkalmazásának lehetőségét és célszerűségét.

A World Wide Web Consortium „HTML5 logo”-t vezetett be a technika népszerűsítésére és egyértelműen bátorítani kezdte a mindennapi használatukat. 2011-ben sor került az első CSS3 modulok ajánlásainak elfogadására, 2012-ben elindult az SVG 2 kidolgozása, a HTML5 szabvány elfogadására pedig 2014 végi határidők tűztek ki.

A HTML5-ben az „5” nem csupán egy verziószám változás, hanem a webes jelölőnyelv legújabb generációjáról van szó. Ugyanakkor visszafelé kompatibilis a korábbi HTML/XHTML verziókkal, és kódolása tekintetében nem revolúció (a múlttal való gyökeres szakítás), hanem evolúció (fokozatos fejlődés). Ha valaki egy régebbi HTML/XHTML verzióban készült weblapja kódjában csak a dokumentumtípus meghatározást átírja, máris HTML5-ös weblapja van – legfeljebb elavult elemeket is használ és nem él a számos új szolgáltatás lehetőségével. Az új szabvány tehát nagyon toleráns és felhasználóbarát, és még a hibás kódok értelmezésére is útmutatást ad a különböző böngészőknek.

Új perspektívát nyitnak a CSS3 által lehetővé tett megjelenítési megoldások, melyek a képszerkesztők és szkript-nyelvek egyes funkcióit átvéve egységesebb szerkezetű és gyorsabban működő weblapokat eredményeznek. Az elkövetkező évben várható a CSS3 egyes új moduljainak (többhasábos elrendezés, transzformációk, animációk, rugalmas dobozok) véglegezése és az SVG 2 különböző megjelenítőméretekhez illeszkedő, fokozott alkalmazása.

A jelenleg zajló mélyreható változások következtében időszerű a korábban megszerzett HTML/XHTML, CSS és SVG ismereteknek az új szabvány szerinti újraértelmezése és bővítése, ill. a most tanulást kezdőknek már az új alapokon való indulása.

A „Szabványkövető statikus honlapok szerkesztése” döntően a World Wide Web Consortium és a Wikipedia publikációi alapján készült, célja egy olyan eszköz nyújtása az érdeklődők kezébe, melynek segítségével programozási ismeretek és elriasztó szakkifejezések nélkül (de mégis a HTML CSS és SVG aktuálisan értelmezett szabályos kódolásával) az idén 25 éves világháló statikus weblapjai készítésének korszerű alapjai remélhetőleg néhány nap alatt elsajátíthatók. Az itt összefoglalt ismeretekkel látványos és szabványkövető webhelyek alakíthatóak ki, melyeket mind a látogatók, mind a keresőprogramok hatékonyan használhatnak.

A webhelyeken felhasználható multimédia tartalmak (kép, hang, animáció, video) létrehozása a téma terjedelménél és eltérő jellegénél fogva nem képezi az összefoglaló tárgyát, az alkalmazások során adottnak vesszük, hogy valamilyen formában már rendelkezésre áll egy ilyen tartalom.

Ebben az átmeneti időszakban (a HTML5.0 „ajánlásra jelölt”, az egyes CSS3 modulok „ajánlás”, „ajánlásra jelölt” és „munkaanyag”, az SVG 2 „munkaanyag” stádiumban vannak) egy rugalmasan reagálni képes elektronikus összefoglaló/ismertető remélhetőleg sikeresen szolgálja az érdeklődők ismeretszerzését.

Budapest, 2014. szeptember

1. ALAPOK

1.1. Tartalom és forma: HTML és CSS

A weboldalak strukturált tartalma HTML jelölőnyelvben (HTML = Hypertext Markup Language = hiperszöveg jelölőnyelv – más fordításban leírónyelv), a formázás és elrendezés CSS stílus(lap)nyelvben (CSS = Cascading Style Sheets = különféle magyar fordításban „lépcsőzetes” vagy „rangsorolt” vagy „egymásba ágyazott” vagy „lépcsőzetesen egymásra épülő” vagy „többszintű” stíluslapok) történő kódolással állítható elő.

A HTML (strukturált tartalom) és CSS (formázás, elrendezés) használatával két olyan - egymáshoz jól illeszkedő és felhasználóbarát - eszköz van a honlapkészítők kezében, melyek jelentősen hozzájárultak a „web” robbanásszerű fejlődéséhez. Alkalmazásuk révén a tartalom és forma egymástól függetlenül változtatható, ugyanahhoz a tartalomhoz több – pl. média-függő - megjelenítés társítható, külső stílusok alkalmazhatók, és használatukkal áttekinthető kódolás és gyorsabb weboldal-letöltés érhető el.

Az SVG (SVG = Scalable Vector Graphics = átméretezhető vektorgrafika) a HTML-el és CSS-el jól együttműködő jelölőnyelv minőségromlás nélkül átméretezhető egyszerű grafikus tartalom előállítására.

A HTML jelölőnyelv, CSS stílus(lap)nyelv és SVG vektorgrafikus jelölőnyelv mindenki számára ingyenesen elérhető és könnyen megtanulható, továbbfejlesztésüket és szabványosításukat a World Wide Web Consortium (röviden W3C) nemzetközi non-profit szervezet koordinálja.

Bár sokan használnak vizuális (WYSIWYG = „what you see is what you get”, magyarra ALAKHŰ-nek fordítható =„Azt Látod, Amit Kapsz, Hűen”) weblapkészítő programokat (kiejtése „vizivig”), melyek alig igénylik a kódolásban való jártasságot, a szabványkövető weboldal építőkockáinak ismerete a megértés és továbbfejlődés szempontjá-ból elengedhetetlen.

A webszerkesztő tanfolyamok általában az Adobe cég Creative Suite (CS) webes grafikus (kereskedelmi) programcsomagjait (a Dreamweaver és a Fireworks/Photoshop/ Flash valamilyen kombinációját) használják.

Mind a HTML, mind a CSS és SVG kódolása a minden operációs rendszerhez ingyenesen mellékelt egyszerű szövegszerkesztő (plain text editor) programmal előállítható. Ezen kívül számtalan ingyenes és kereskedelmi, különböző funkciókat támogató szerkesztőprogram van forgalomban. A lényeg azonban nem egy adott programnak, hanem a három nyelv kreatív módon való, önálló használatának az elsajátítása.

A szabványos kódolás alkalmazásának (az időtállóság, a jobb áttekinthetőség és köny-nyebb továbbfejleszthetőség mellett) az a célja, hogy egy adott weblapot minden böngésző lehetőleg ugyanolyan módon jelenítsen meg. Miután tendenciájukban a böngészők a szab-ványok felismerésének és azonos értelmezésének irányába fejlődnek, a szabványkövető kódo-lás a leghatékonyabb módja a tervező szándéka szerinti, időtálló és egységes megjelenítésnek.

Az itt tárgyalt HTML-el, CSS-el és SVG-vel alapvetően statikus weblapok készíthetők, a dinamikus /interaktív weboldalak a statikus struktúrába ágyazott egyéb elemekkel (szkript-nyelvekkel) valósíthatók meg. A szkript-nyelvek tárgyalása meghaladja a „Szabványkövető statikus honlapok szerkesztése” kereteit, azonban néhány dinamikus hatás az új CSS3 tulajdonságokkal is kialakítható.

1.2. A weblapkészítés hardver és szoftver igénye

A weblapkészítés alapjainak megtanulásához és statikus weblapok szerkesztéséhez nincsen szükség speciális műszaki háttérre vagy pénzügyi befektetésre, de még állandó internetkapcsolatra sem. Egy egyszerű editor-t már eleve minden mikroszámítógéphez adnak (nem javasolt viszont a formázott - rich text - szöveget előállító programok - pl. WordPad, Word, stb. - használata a HTML/CSS/SVG kódoláshoz !), és a legismertebb böngészők ingyenes letöltése után a tanulás és/vagy weblapszerkesztés off-line üzemben is történhet.

Hasznos lehet a Windows-os PC-k Notepad-je (Jegyzettömb) helyettesítésére kifejlesztett Notepad++ editor (aktuális verziója 6.6), illetve a kész weblap szerverre való feltöltéséhez a Total Commander fájlkezelő (aktuális verziója 8.5) is, melyek ingyenesek, hatékonyak, és mindkettő magyar nyelven is letölthető.

A weblap megjelenítését a böngésző hajtja végre, mely egy adott operációs rendszerrel működik együtt. A jelenleg használatos főbb operációs rendszer családok és főbb böngészőik:

a) desktop/laptop (x86 processzoros)eszközök

Microsoft Windows

Megnevezés	Forgalmazás kezdete	Böngészői	Forgalmazásuk kezdete
XP (experience)	2001 október	Internet Explorer 6	2001 augusztus
		Internet Explorer 7	2006 november
		Safari for Windows 5.1	2011 július
		Internet Explorer 8	2009 március
		Firefox, Chrome, Opera	hat hetente frissülnek
Vista	2007 január	Internet Explorer 8	2009 március
		Internet Explorer 9	2011 március
		Firefox, Chrome, Opera	hat hetente frissülnek
7	2009 október	Internet Explorer 8	2009 március
		Internet Explorer 9	2011 március
		Internet Explorer 10	2013 február
		Internet Explorer 11	2013 november
		Firefox, Chrome, Opera	hat hetente frissülnek
8	2012 október	Internet Explorer 10	2012 október
		Firefox, Chrome, Opera	hat hetente frissülnek
8.1	2013 október	Internet Explorer 11	2013 október
		Firefox, Chrome, Opera	hathetente frissülnek

Megjegyzések:

- az Internet Explorer 6 és Internet Explorer 7 használata már marginálisra csökkent
- az Apple 2008-től *Safari for Windows*–t is forgalmazott, ennek fejlesztése az 5.1 verzióval megszűnt (ez még letölthető az Apple honlapjáról XP, Vista és Windows 7-re, de használata elenyésző)
- a Windows-os felhasználók többsége nem Internet Explorer-t, hanem *Firefox for Windows*-t, *Chrome for Windows*-t, *Opera for Windows-t*, ill. „kis böngészők”-et (Avant Browser, Comodo Dragon, K-Meleon, Lunascape, Maxthon, Rockmelt, SRWare Iron, stb.) használ
- sajátos megoldás a Google *Chrome Frame*-je: amennyiben a Windows-os felhasználó letölti, az az Internet Explorer alatt fut, így az IE-t használva az új HTML5/ CSS3 funkciók és tulajdonságok is értelmezettek és megjeleníthetők lesznek. 2014 elejétől felhagyott a Google a fejlesztésével, mivel úgy véli, hogy a HTML5/CSS3-at nem támogató Internet Explorer-ek aránya annyira visszaszorul, hogy nem érdemes külön terméket fenntartani a kiegészítésükre.

Apple (Mac) OS X

Az X egyrészt arra utal, hogy ez az Apple tizedik operációs rendszer családja (az X mint római 10-es), másrészt a Unix-ra, melynek alapjaira épül.

Verziószám	Megnevezés	Forgalmazás kezdete	Processzor	Böngészője
10.0	Cheetah	2001. március	POWER PC	IE for Mac
10.1	Puma	2001. szeptember	POWER PC	IE for Mac
10.2	Jaguar	2002. augusztus	POWER PC	IE for Mac, Safari 1.0
10.3	Panther	2003. október	POWER PC	IE for Mac, Safari 1.3
10.4	Tiger	2005. április	POWER PC és Intel	Safari 4.1
10.5	Leopard	2007. október	POWER PC és Intel	Safari 5.0
10.6	Snow Leopard	2009. augusztus	Intel	Safari 5.1 Chrome, Firefox, Opera
10.7	Lion	2011. július	Intel	Safari 6.1 Chrome, Firefox, Opera
10.8	Mountain Lion	2012. február	Intel	Safari 6.1 Chrome, Firefox, Opera
10.9	Mavericks	2013. október	Intel	Safari 7.0 Chrome, Firefox, Opera
10.10	Yosemite	2014. szeptember	Intel	Safari 8.0 Chrome, Firefox, Opera

Az operációs rendszer első négy verziója az Apple és az IBM által közösen kifejlesztett POWER PC (*Performance Optimized With Enhanced RISC Performance Chip*) processzszorokon futott, két verzió mindkét processzorra készült, majd teljesen áttértek az Intel-re. A 10.8-tól elmaradt a megnevezésből a *Mac* szó, a verziószám mellett a ragadozó nagymacskákat a 10.9-től kaliforniai helységnevek váltották fel.

Megjegyzések:

- A Microsoft IE 5.5-el megszűntek az Internet Explorer *Mac*–es böngészőverziói
- A Safari a 6.0-tól külön nem tölthető le, csak az újabb operációs rendszerrel vagy software update-ként adják a Safari böngészőket az OS X rendszerekhez
- A *Mavericks (OS X 10.9)* a *Safari 7.0*-val elvileg a *10.6 Snow Leopard*-ig visszamenően ingyenesen letölthető, de a gyakorlatban sok esetben problémákat okozott, ezért sokan nem éltek a lehetőséggel
- A Mac OS X-es felhasználók számottevő része *Chrome for Mac* (ill. kis részben *Firefox for Mac* és *Opera for Mac*) böngészőt használ. A Chrome 21-től, a Firefox 17-től már csak a Mac OS X 10.6-al és az annál újabb operációs rendszerekkel használhatóak.

Linux

A Chrome és Firefox a *Windows*-os és *Mac*-es kiadásokkal szinkronban rendszeresen *Linux*-os verziókkal is megjelenik. Az Opera a 12.1-es kiadást követően (*Presto*-ról *Blink*-re váltás) egy ideig nem készített Linux-ra böngészőt, bő egyéves szünet után, várhatóan az Opera 25–től ismét megjelenik a Linux-os verzió is.

b) mobil (ARM processzoros) eszközök

Microsoft Windows RT

Tabletekre készült, az asztali Windows programokat nem futtató operációs rendszer. Jelenleg csak a Microsoft és a nemrég tulajdonába került Nokia Mobile forgalmaz Windows RT-s tabletet, és az alkalmazásokkal való ellátottsága is szerény. Microsoft bejelentése szerint az asztali, tablet és okostelefonos operációs rendszerüket egységesíteni fogják.

Microsoft Windows Phone (WP)

Verziószám	Forgalmazás kezdete	Böngésző
Windows Phone 7.0	2010 október	Internet Explorer Mobile for WP (IE 8 alapon)
Windows Phone 7.5	2011 október	Internet Explorer Mobile for WP (IE 9 alapon)
Windows Phone 7.8	2013 február	Internet Explorer Mobile for WP (IE 9 alapon)
Windows Phone 8	2012 október	Internet Explorer Mobile for WP (IE 10 alapon)
Windows Phone 8.1	2014 április	Internet Explorer Mobile for WP (IE 11 alapon)

Az operációs rendszer alapján a mai okostelefonos Windows-tábor kettészakadt 7.x és 8.x-re (a 7.x ugyanis nem frissíthető tovább 8.x-re). A 7.x fejlesztése tavaly megszűnt, a 8.x felhasználók vannak már túlsúlyban. Microsoft bejelentése szerint az asztali, tablet és okostelefonos operációs rendszerüket egységesíteni fogják.

Google Android

Verziószám	Megnevezés	Forgalmazás kezdete
Android 1.0	Apple pie	2008 október
Android 1.1	Banana bread	2009 február
Android 1.5	Cupcake	2009 április
Android 1.6	Donut	2009 szeptember
Android 2.0	Eclair	2009 október
Android 2.1	Eclair	2010 január
Android 2.2	Froyo	2010 május
Android 2.3	Gingerbread	2010 december
Android 3.0	Honeycomb	2011 február
Android 3.1	Honeycomb	2011 május
Android 3.2	Honeycomb	2011 július
Android 4.0	Ice Cream Sandwich	2011 október
Android 4.1	Jelly Bean	2012 június
Android 4.2	Jelly Bean	2012 október
Android 4.3	Jelly Bean	2013 július
Android 4.4	KitKat	2013 október

Az Android-oknak (az operációs rendszernek megfelelő verziószámú, fokozatosan bővülő HTML5/CSS3 funkciókkal rendelkező) saját Android böngészőjük van. A 4.0-tól használható az asztali verziókkal együtt 6 hetente frissülő *Chrome for Android* böngésző is, melynek használata hamarosan utoléri az Android-okét. A *Firefox for Android* szintén a Firefox asztali böngészővel együtt frissül az Android 2.2 verziótól újabb operációs rendszerekhez.

Mivel az Android 2.1 és annál korábbi verziók már eltűntek, a 3.x pedig el sem terjedt, az Android-os tábor az operációs rendszer alapján kettészakadt 2.2/2.3 (okostelefonos) ill. 4.x (okostelefonos és tabletes) csoportra. Az előbbinek a *Firefox for Android*, az utóbbinak a

Chrome for Android és *Firefox for Android* (ill. a saját Android böngésző) révén az új HTML5/CSS3 funkciókat kezelő böngészőkkel való ellátottsága elvileg biztosított.

Az Android-okhoz használt további ismertebb böngészők az *Opera Mobile*, a *Dolphin HD/Mini* és a *Boat*.

Apple iOS

2007 júniusában az első iPhone-al jelent meg az 1.0 verzió (melyet még *iPhone OS*-nek hívtak), 2008 júliusában a 2.0, 2009 júniusban a 3.0 verziót adták ki. Az operációs rendszer csak az Apple mobil eszközein (*iPhone, iPod Touch, iPad*) használható, az egyes típusokon alkalmazható legfrissebb verziókat az alábbi táblázat mutatja:

eszközök	a legújabb használható iOS/SafariMobile	iOS/SafariMobile kiadás dátuma
iPhone (eredeti), iPod Touch (eredeti)	3.1	2010 február
iPhone 3G, iPod Touch 2	4.2	2010 november
iPod Touch 3, iPad (eredeti)	5.1	2012 május
iPhone 3GS, iPod Touch 4	6.1	2013 január
iPhone 4	7.1	2014 július
iPhone 4S, iPhone 5, iPhone 5C, iPhone 5S, iPhone 6, iPhone 6Plus, iPod Touch 5, iPad 2, iPad Retina, iPadAir, iPad Mini, iPad Mini/Retina	8.0	2014 szeptember

A fenti operációs rendszerekhez azonos verziószámmal csatlakoznak az alapértelmezett SafariMobile (*iOS Safari*) böngészők. Az iOS tábor az iOS/SafariMobile 7.1 és 8.0 ingyenes és visszamenőleges (operációs rendszer és böngésző szerinti) frissítéssel viszonylag homogén marad (bár a frissítéssel műszaki okokból sokan nem élnek).

A Safari Mobile alternatívájaként használhatók a *Chrome for iOS*, *Atomic* és *Dolphin* böngészők is.

BlackBerry OS

A RIM (Research in Motion) 1999-ben pager/e-mail klienséhez dobta piacra a *Blackberry OS* első verzióját, mely a 3.x-től (2002) használható okostelefonhoz. A 4.0 (2003 június), 5.0 (2008 augusztus) és 6.0 (2010 augusztus) verziók már nem frissíthetők. A 2011 augusztusi 7.0 (ill. a 2012 januári 7.1 upgrade-je) és a 2013-as *Blackberry OS 10* az élő operációs rendszerek, melyek saját böngészővel rendelkeznek. 2013 elején maga a cég is a Blackberry nevet vette fel a RIM helyett.

Mozilla Firefox OS (FFOS, FxOS, korábban B2G)

2011 nyarán jelentette be a Mozilla, hogy B2G néven (*B2G = Boot to Gecko*) nyílt forráskódú, ARM processzorokon futó, webes technológiákon (HTML5, CSS3 és JavaScript) alapuló operációs rendszert fejleszt. 2013 februárjában mutatták be a barcelonai Mobile World Expo-n immár Firefox OS néven, mely a Firefox böngészők Gecko böngészőmotorján, a Gaia felhasználói felületen és a Gonk Linux-kernelen alapul. Kereskedelmi forgalomba

2013. július 1-től kerültek az 1.0 verziót futtató, belépő szintű okostelefonok. Az 1.0 a Gecko 18.-on, az 1.1 a Gecko 23-on, az 1.2 a Gecko 26-on, az 1.3 a Gecko 28-on, az 1.4 a Gecko 30-on alapul - ezután Gecko 32 alapon 2.0-ra ugrik a verziószám. A tervek szerint negyedévente (a páros számú böngészőverzió számokkal) jelennek meg az új Firefox OS verziók, hogy utol tudják érni a versenytársakat. Forgalmazása első évében 5 szolgáltató 15 (latin-amerikai és közép-kelet- ill. dél-európai) országban vezetett be Firefox OS-el működő telefonokat, a tömeges elterjedést 2014 második félévétől az indiai (és további 5 latin-amerikai ország) alsó árkategóriájú piacától várják.

c) piacról kivonuló mobil operációs rendszerek

A Microsoft *Windows Mobile* operációs rendszere piacvezetőnek indult 2003-ban okostelefonra és zseb-PC-khez, 2007-ben volt a csúcson 42%-os piaci részesedéssel - a Nokia és Apple kivételével valamennyi jelentős gyártó forgalmazta - de 2010-re az Android és iOS dominancia következtében 5% körülire esett a részesedése, a gyártók – szintén a Nokia és Apple kivételével - Androidra váltottak. Az operációs rendszer családtagjai: Windows Mobile 2003 (2003), Windows Mobile 2003 SE (Second Edition, 2004), Windows Mobile 5.0 (2005), Windows Mobile 6.0 (2007), Windows Mobile 6.1 (2008), Windows Mobile 6.5 (2009). 2010-től az új alapokra épült, vele nem kompatibilis *Windows Phone* mobil operációs rendszer család váltotta fel a *Windows Mobile 6.x*-eket.

A *WebOS*–t eredetileg a Palm fejlesztette ki 2009-ben *Palm WebOS* néven okostelefonokra és tabletekre. Felvásárolta a Hewlett-Packard, majd 2012 elején, miután megszüntette a mobil eszközeinek gyártását, a 3.0 verzióig jutott operációs rendszert *Open WebOS* név alatt nyílt forráskódúvá tette. Jelenleg az LG használja az okos tv készülékeiben.

A *Symbian*-t a ’80-as évek végétől a Psion kezdte fejleszteni *EPOC* néven kézi számítógépekhez – az 5-ös verzióig jutva el vele. 1998-ban a Nokia-val, Motorola-val és Ericsson-nal létrehozták a Symbian Software konzorciumot, melyhez egy évvel később a Panasonic, Siemens és Sony-Ericsson is csatlakozott. Az első okostelefon a 6-os verzióval – immár *Symbian OS*-ként - 2000-ben (Ericsson), ill. 2001-ben (Nokia) debütált. A Nokia telefonjaival először piacvezetővé vált, majd 2010-től a konkurensekkel szemben versenyhátrányba került. 2011-től az Accenture-hez szervezték ki az immár nyílt forrásúvá tett operációs rendszer (2016-ig történő) támogatását, a *Symbian*-os okostelefonok gyártását pedig 2012 végével beszüntették.

d) böngészők frissítése

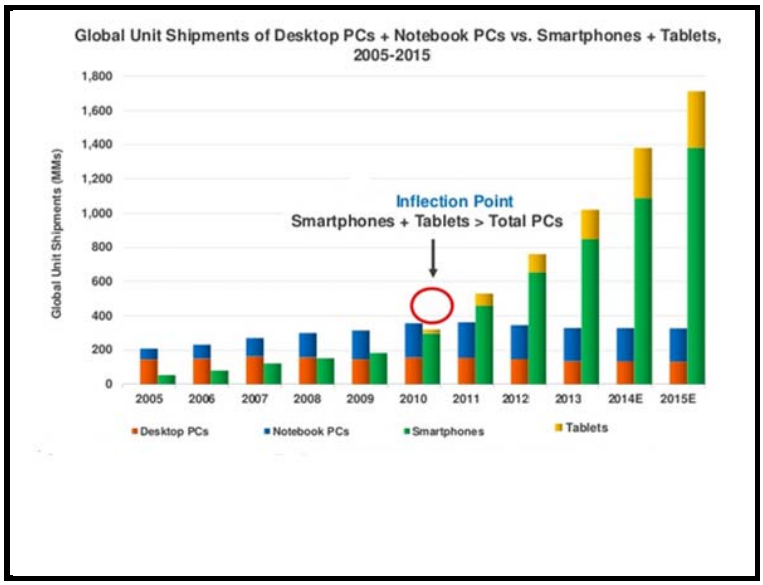
A Chrome, Firefox és Opera asztali és mobil verzióit egyaránt 6 hetente adja ki. A Firefox úgynevezett ESR (Extended Support Release = kiterjesztett támogatású kiadás) verzióval is rendelkezik azok számára, akik nem kívánják követni a gyakori frissítést – ezt 54 hétig automatikusan ellátják biztonsági frissítésekkel, az időközben bevezetett új funkciókkal viszont nem. Az ESR kiadás csak asztali böngészőkhöz készül, mobil ESR nincsen.

A Safari és az Internet Explorer új asztali és mobil verziói az operációs rendszerek frissítésével együtt (kb. egy – másfél évenként) jelennek meg (ez jelentős pozitív változás az IE korábbi sokéves ciklusához képest).

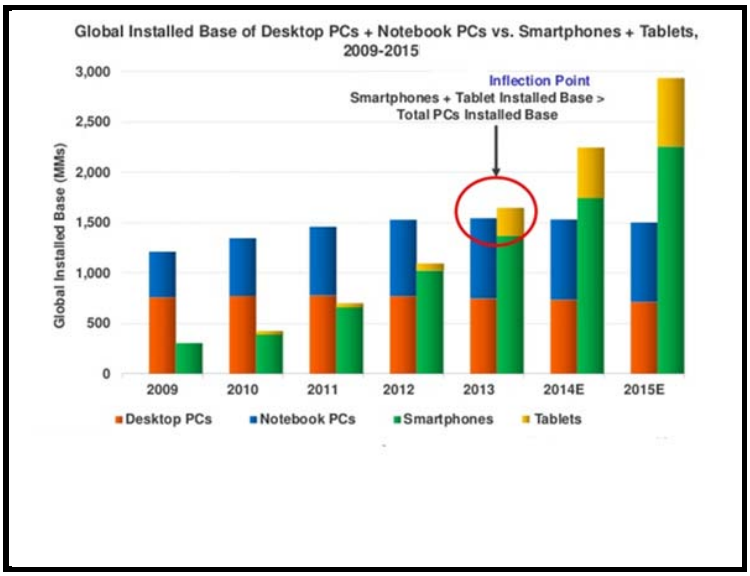
A legújabb böngészők automatikus frissítésre vannak beállítva. Aki nem biztos benne, hogy milyen böngészőverzió van telepítve a gépére, pl. a www.fmbip.com weboldalt felkeresve megtudhatja.

e) weblap szerkesztésnél figyelembe veendő tendenciák

Az ARM processzoros okostelefonok és tabletek értékesítése 2010 végén meghaladta a x86 processzoros asztali/hordozható gépekét. Az ARM mobil platformok terjedése a belátható jövőben továbbra is meredeken nőni fog, míg a x86-os desktop/laptop-ok termékéletciklusuk érett szakaszába értek:



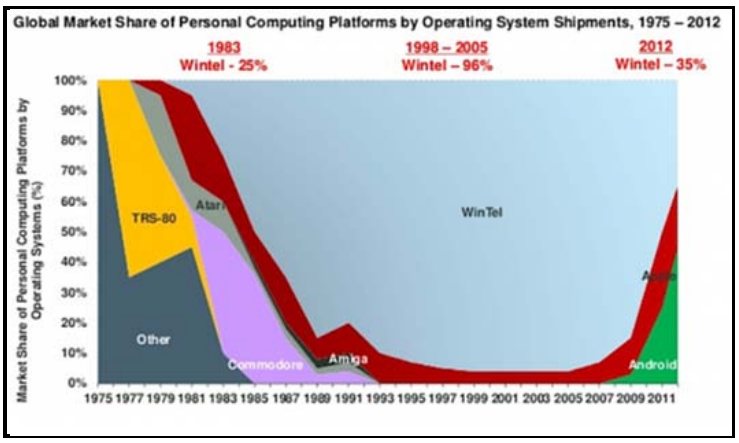
A fenti értékesítési tendenciák következtében az ARM mobil platformos készülékek üzemelő állománya 2013 közepén meghaladta a desktop/laptop-okét:



A mobil eszközállomány növekedése – fáziskéséssel bár – de fokozatosan maga után vonja az internetes adatforgalom egyre növekvő részének mobil platformokra történő áthelyeződését. Magyarországon 2014 nyarán a magyar weblapok letöltésének kb.13-14%-a történik mobil készülékről.

A mikroszámítógép rendszerek „hőskorszakbeli” sokszínűsége a ’90-es elejére megszűnt, a x86 processzorokon futó Windows-ok részesedése évekig a 95%-ot is meghaladta. A

mobil eszközök terjedésével ismét sokszínűvé válik az Internethez kapcsolódó operációs rendszerek száma:



Az előbbiekben felsoroltak mellett számos új platform is teret nyerhet (pl. mobil Ubuntu Linux, Tizen, Sailfish operációs rendszerek). Ezért különösen fontos lehet a HTML5 mint platformfüggetlen alkalmazás elterjedése.

1.3. Statikus webhelyek kialakításának szokásos módja

A számítógépben egy külön (al-mappákat nem tartalmazó) mappában célszerű a web-oldalak valamennyi alkotóelemének - a tartalmat hordozó HTML oldalaknak, a formázást adó CSS stíluslap(ok)nak, a felhasználandó multimédiás elemeknek - a gyűjtése.

A gyűjtőmappa (*site root*) neve bármi lehet (kiterjesztése nincsen). A webhely kezdő HTML-fájljának neve mindig *index*, a többi oldal fájlneve lehetőleg a tartalmukra utaló, ékezet nélküli kisbetűs szó legyen, és valamennyi fájlnév kiterjesztése *.html*. A böngészők mindig az *index.html* nevű oldalt fogják a webhely kezdő- vagy címlapjának (home page) tekinteni. A stíluslap(ok)nak *.css* kiterjesztésű ékezet nélküli kisbetűs szó legyen a neve.

Megjegyzés: Néhány régi szerkesztőprogram *.html* helyett *.htm* kiterjesztést használ. Ez ugyanolyan jó, de akkor következetesen mindenhol *.htm*-et kell alkalmazni.

A weblapkészítés ill. annak tanulása két úton történhet:

- vagy először a HTML kódolás (a strukturált tartalom) előállítása, ill. annak megtanulása, aztán a kész tartalom CSS formázása, ill. a CSS megtanulása

- vagy a strukturált tartalom részenkénti előállítása és az egyes részek menet közbeni formázása, tanulás esetén pedig felváltva a HTML és CSS kódolás használatának elsajátítása.

Gyakorlott weblapkészítőknél az egyéni ízlés dönt, tanulásnál pedig mindkét sorrendnek van előnye és hátránya. Először a HTML-kódolást majd utána a CSS-t külön-külön tanulva - egy-egy zárt szabályrendszerben mozogva - könnyebb kiigazodni és előrehaladni. Ugyanakkor a HTML-lap önmagában nem mutatós, látványos eredményt és sikerélményt csak a CSS-el történő formázásával lehet elérni.

Mindkét esetnek úgy lehet egyidejűleg eleget tenni, ha külön-külön történik a HTML és CSS tárgyalása, ugyanakkor a HTML-részben az egyes témák végén utalás van egyes vonatkozó CSS fejezetekre - így mindenki maga döntheti el, hogy melyik módszert találja célravezetőbbnek, milyen sorrendben halad.

A HTML mindig az a tartalmi alap, amihez a CSS formázás hozzárendelhető, tehát mindjárt a CSS-el kezdeni nem lehet. A nagyszámú variációs lehetőségek miatt ugyanakkor a „nagyobb falat” a CSS megtanulása, ill. a formázás kialakítása. Az SVG-t a HTML és CSS után célszerű tanulni, mert így csupa ismert fogalommal találkozik az ember.

Megjegyzés: A böngészők frissülő verziói folyamatosan vezetik be az új HTML5/CSS3 és SVG 2 funkciókat. Az elkészült honlapot a főbb böngészőkkel (Chrome, Firefox, Internet Explorer, Safari, Opera) ellenőrizni kell, hogy nincs-e még olyan eltérés valamelyik funkció értelmezésében, ami valamelyikük esetében problémát okozna a megjelenítésben.

A könnyű megjegyezhetőség érdekében a tárgyalt kódolások a Chrome 35, Firefox 30, Opera 24, Internet Explorer 11 és Safari 7 verziók figyelembevételével történtek. A Safari-val csak Apple géppel rendelkezők tudják a leírtak szerint elkészült honlapokat ellenőrizni (új Windows-os vagy bármilyen Linux-os verziója nem létezik), de az ismertett témák esetében általában jó egyezést mutat a Chrome-al.

2. HTML5

2.1. A HTML nyelvtana

2.1.1. Címkék és elemek

A HTML jelölőnyelvben a megjelenítendő tartalomba **címké**et (angolul *tag*=címke) helyezünk el, melyek megszabják, hogy a böngészők hogyan értelmezzék a megjelölt tartalmi részeket. A címkék megnevezése egyben kifejezi (angolul) a tartalmukat is. Az angol szó vagy rövidítés mindig „<” és „>” („kisebb mint”, ill. „nagyobb mint” = „*less than*”, „*greater than*”, vagy másképpen „csúcsos zárójel”= „*angle bracket*”) jelek között helyezkedik el. Elvileg kisbetű/nagybetű érzéketlenek, de csupa kisbetűt célszerű használni (részben az évtizedes XHTML-es hagyomány, részben a kisbetű-nagybetű keveredésből származó hibaforrások elkerülése miatt).

A címkék nagy többsége párban, kezdő- és zárócímkeként *(start tag, end tag*) szerepel, a kezdő- és zárócímke közötti különbség annyi, hogy ami kezdőcímkeként <…>, az zárócímkeként </…>, azaz egy „per” vagy „törtvonal” (*slash*) karakterrel bővül.

A HTML-ben kódolt tartalom páros címkékkel tehát így néz ki:

```
<valamilyen páros címke>
    tartalom egy része
</valamilyen páros címke>
<másik páros címke>
  tartalom további része
</másik páros címke>
…….stb.
```

Páratlanul csak néhány címke szerepelhet, ezek használatakor a kódolt HTML tartalom:

```
<valamilyen páros címke>
    tartalom egy része
</valamilyen páros címke>
<egy páratlan címke>
<másik páros címke>
 tartalom másik része<másik páratlan címke>ismét tartalmi rész
</másik páros címke>
 ……….stb.
```

A páros címkék kezdő- és zárócímkéi által bezárt tartalmi részek az elemek. A fenti sémából tehát egy elem a:

```
<valamilyen páros címke>
   tartalom egy része
</valamilyen páros címke>
```

Egy másik elem a:

```
<másik páros címke>
 tartalom másik része<másik páratlan címke>ismét tartalmi rész
</másik páros címke>
```

A páratlan címkék üres elemet alkotnak, tehát üres elem pl. a

```
<egy páratlan címke>
```

és a

```
……<másik páratlan címke>……
```

A fentiekből látszik, hogy a címkék (elemek) egymásba is ágyazhatók, azaz a páros címkék közrefoghatnak páratlan címkéket (vagy további páros címke-párokat is).

A kódolás áttekinthetőségét javítja, ha az egyes elemeket - mint azt az előzőekben is tettük - új sorokban kezdjük és zárjuk le (ez tartalmi szempontból azonos a folyamatosan írt kóddal, de a hibakeresést, módosítást és kiegészítést nagyban megkönnyíti).

Megjegyzés: Az üres elemeket létrehozó páratlan címkék száma sokkal kisebb, mint a nem üres elemeket létrehozó párosaké, de szerepük ugyanolyan nélkülözhetetlen; ezek definiáltak meta-adatokat, link-eket, sortörést, szótörést, állóképeket, külső weblaptartalmakat, vízszintes elválasztó vonalat, stb.

2.1.2. Jellemzők és értékek

Egy elemnek lehet egy vagy több **jellemzője** (*attribute* = attributum, jellegzetes tulajdonság) és a jellemzőnek van **értéke** (*value* = érték) - ezek az adott elemhez valamilyen járulékos információt adnak meg. A jellemző („tulajdonság”-nak is fordítják magyarra, de miután a CSS-nél is lesz „tulajdonság”, és ott más angol szót használnak erre, érdemes a magyarban is különbséget tenni) a kezdő címkébe, a címke után egy betűközt kihagyva írható be, az érték pedig =”…” formában adandó meg.

Ha definiálunk valamilyen jellemzőt, akkor értéket is kell adni neki – kivétel az a néhány eset, amikor a tulajdonság értéke logikai „igaz”, vagyis csak egyszerűen deklarálunk valamit. Az érték nélkül is megadható néhány jellemzőt az alkalmazásuk során jelezzük majd.

A HTML-ben kódolt tartalom tehát jellemzőkkel és értékekkel is rendelkező címkékkel:

```
<valamilyen páros címke valamilyen jellemző=”egy érték”>
    tartalom egy része
</valamilyen páros címke>
<másik páros címke akármilyen jellemző=”másik érték”>
  tartalom további része
</másik páros címke>
……stb.
```

Egy elemhez több jellemző is definiálható, a jellemzők megadási sorrendje közömbös, és üres betűköz választja el ilyenkor egymástól a jellemző-érték párokat:

```
<valamilyen páros címke jellemző1="érték1" jellemző2="érték2">
    tartalom egy része
</valamilyen páros címke>
<másik páros címke jellemző3="érték3" jellemző4="érték4">
  tartalom további része
</másik páros címke>
…….stb.
```

Páratlan címkékhez (üres elemekhez) hasonló módon megadható egy vagy több jellemző - értelemszerűen a kezdő címke itt magának az egyetlen, páratlan címkének felel meg:

```
<egy páratlan címke jellemző5="érték5" jellemző6="érték6">……
……<másik páratlan címke jellemző5="érték5">……
```

A jellemzők és értékek csak az adott konkrét elemre vonatkoznak, és páros címke esetén a zárócímkével, ill. páratlan címke esetén a záró csúcsos zárójellel érvényüket vesztik.

2.1.3. Strukturált tartalom

Egy weboldal strukturált tartalmának HTML-kódolással történő létrehozása a HTML-címkéknek és esetleges jellemzőiknek/értékeiknek a tartalomba való behelyezéséből áll. Ezt az elemi kockákból felépített logikai szerkezetet (strukturált tartalmat) a HTML-címkék alapján a böngészők értelmezik és megjelenítik.

Az egyes (páros és/vagy páratlan) címkék – jellegükből adódóan - sorban egymás mellé (*inline*) vagy blokkszerűen egymás alá (*block*) helyezik el az általuk létrehozott elemeket. A megjelenítéssel kapcsolatos minden további feladatot a CSS stílus(lap)nyelv lát el.

2.2. A weboldal szerkezete

Minden weboldal 3 fő részből áll:

a) dokumentumtípus meghatározás

b) fej

c) törzs

a) A **dokumentumtípus meghatározás** (DTD = Document Type Definition) adja meg a weboldal nyelvtanát a böngésző számára. A megadás módja:

```
<!doctype html>
```
(doctype = *document type* = dokumentum típus)

Ez biztosítja, hogy a böngésző szabálykövető üzemmódban (*standard mode, no quirks mode*) értelmezze a HTML weboldalt. Ha hiányosan, hibásan, vagy egyáltalán nincs megadva a dokumentumtípus, akkor a böngésző trükköző üzemmódba (*quirks mode*) válthat, és magától próbálhatja értelmezni a kódolást – ez természetesen magában hordozza a hibás megjelenítés lehetőségét.

Megjegyzés: Régebben a *doctype*-ot csupa nagybetűvel írták. Tekintettel arra, hogy a címkék nem kisbetű/nagybetű érzékenyek, most is szabályos akár a csupa nagybetűs, akár a *Doctype*, de még a doCTYpe írásmód is. Legegyszerűbb azonban a csupa kisbetűk használata.

A dokumentumtípus meghatározás után egy *<html>* és *</html>* páros címkék közötti rész alkotja a böngésző által értelmezett és megjelenített weblapot. A böngésző számára a *<html>* kezdőcímkénél kezdődik, és a *</html>* zárócímkénél ér véget a megjelenítendő weboldal.

A keresőprogramok hatékonyságát elősegítendő - „helyes gyakorlat” címén - javasolt a honlap fő nyelvének megadása is, mely magyar nyelvű weblapnál a kezdő *html* címke *lang=”hu”* (*language*=nyelv, Hungarian=magyar) jellemző/érték kiegészítését jelenti (*„en”* az angol nyelv, *„fr”* a francia, stb. – még az amerikai és brit angol nyelvhasználat között is különbség tehető az *„en-US”* ill. *„en-GB”* értékekkel).

Megjegyzés: Valamennyi nyelv kódját az IETF BCP 47 dokumentum tartalmazza (IETF=Internet Engineering Task Force, BCP=Best Current Practice, a dokumentum címe „Tags for Identifying Languages”).

A magyar nyelvű weblap szerkezete tehát:

```
<!doctype html>
<html lang="hu">
     fej
    törzs
</html>
```

b) A **fej** (*head* =fej) az adott HTML-oldalra vonatkozó meta-adatokat tartalmazza. A meta-adatok a fájllal kapcsolatos különféle jellemzők és utasítások tárolására szolgálnak. Néhány meta-adatnak saját címkéje van (pl. *title*=cím, *link*=kapocs, *style*=stílus) a többiek a *meta* címke jellemzőjeként adhatók meg. A meta-adatokra részletesebben a HTML-rész végén visszatérünk, itt elegendő csak a karakterkészlet, a weblap címe és a CSS stílus megadásának ismertetése.

A fej HTML kódolása: (fontos a meta-adatok sorrendjének betartása !)

```
<head>
  karakterkészlet megadása
  weblap címe
  külső stíluslap csatolása
  beágyazott stílus megadása
</head>
```

b/1) **A karakterkészlet megadása*:***

```
<meta charset="utf-8">
<meta charset="iso-8859-2">
```

vagy

meta a címke, *charset* (charset = character set = karakterkészlet) a jellemző, *utf-8* vagy *iso-8859-2* az érték. A *meta*-nak nincsen zárócímkéje, egyedül használandó (páratlan címke).

Az Unicode Consortium non-profit szervezet kidolgozott egy olyan kódkészletet, mely az összes élő, halott és mesterséges kultúra írásjeleinek és egyéb kiegészítő karaktereknek az ábrázolására alkalmas. Legfrissebb a 2014. júniusában kiadott 7.0-s szabvány, mely 113 021 karaktert tartalmaz - a 8.0-s változat várhatóan 2015 nyarán jelenik meg – de a verziószámot nem kell definiálni a *charset*-ben. Az *UTF-8* a Universal Character Set Transformation Format (Univerzális Karakterkészlet Átalakítási Formátum) 8-bites változata.

Az UTF-8 kódkészlet alkalmazása tipikus magyar szöveg esetén a Latin2-höz képest kb. 10% fájlméret növekedést okoz (angol nyelvű szöveg UTF-8 ábrázolásának és Latin1 ábrázolásának fájlmérete megegyezik).

A HTML5 minden nyelven az UTF-8 alkalmazását javasolja !

Megjegyzés: A Windows XP előtti operációs rendszerek nem ismerik az UTF-8-at, és több régebbi webszerkesztő program is az ISO-8859 szerinti karakterkódolást ajánlja fel alapértelmezettként. Magyar nyelvű weblapoknál az ISO 8859-2 (Latin2 vagy közép-európai) készletet kell ilyenkor beállítani (a 8859-1 vagy Latin1 a nyugat-európai és amerikai, az 8859-3 a dél-kelet-európai, a 8859-4 a skandináv és balti, a 8859-5 a cirill, a 8859-6 az arab, a 8859-7 a görög, a 8859-8 a héber karakterkészlet, stb.). Az ISO-8859-2 készlet által nem tartalmazott karaktereket a későbbiekben említettek szerint, kódolva lehet megadni.

Figyelem! A hibásan megadott karakterkészlet (elsősorban az ékezetes betűknél) téves megjelenítést idézhet elő. Ha nincs karakterkészlet megadva, az a karaktertévesztés mellett a weblap biztonsági sebezhetőségét is okozhatja.

A speciális magyar karakterek helyes megjelenítésének ellenőrzésére kialakult néhány olyan kifejezés, mely az ékezetes magánhangzók mindegyikét egyszer tartalmazza, pl.:

tüskéshátú kígyóbűvölő, öt szép szűzlány őrült írót nyúz, árvíztűrő tükörfúrógép, stb.

Ilyeneket bárki szabadon kitalálhat, és az sem baj, ha egy ékezetes magánhangzó egynél többször fordul elő (pl. *„Üdvözöljük számítógépünk képernyőjén!”*, stb.).

Az angol nyelvben a *„The quick brown fox jumps over the lazy dog”* (A gyors barna róka átugorja a lusta kutyát) az összes angol betűt tartalmazó szokásos ellenőrző szöveg.

b/2) **A weblap címének megadása:** (title =cím)

```
<title>…….</title>
```

A cím tartalmazhat magyar ékezetes szavakat és nagybetűket is. A tartalmat jól kifejező cím segíti a keresők helyes találatát. A cím a böngésző címsorában, ill. a keresők találatainak felsorolásában és a Kedvencek/Könyvjelzők menüjében jelenik meg.

b/3) **Külső CSS stíluslap(ok) csatolása*:***

```
<link rel="stylesheet" href="…….css">
```

A link (kapocs) a címke, *rel* (relation = viszony) és *href* (*hypertext reference* =hiperszöveges hivatkozás) a jellemzők, *stylesheet* (stíluslap) és a *.css* kiterjesztésű CSS fájl neve az értékek. A *link*-nek nincsen zárócímkéje, egyedül használandó (páratlan címke).

A *rel* jellemző jelzi a böngészőnek, hogy a hivatkozott (CSS) dokumentum milyen viszonyban van a hivatkozó (HTML) dokumentummal. (A megadott érték szerint stíluslap viszonyban van vele.)

A *href* jellemző annak a CSS stíluslapnak a fájlnevét és helyét (elérhetőségi útját) adja meg, melyet a HTML dokumentumhoz akarunk kapcsolni. Miután egy mappába gyűjtjük a HTML lapot és a CSS stíluslapot, elég csak a fájlnevet és kiterjesztését megadni. A név szabadon választható, de célszerűen utaljon a tartalmára, kisbetűs, ékezetek és szóköz nélküli legyen. A kiterjesztés kötelezően *.css*.

b/4) **Belső vagy beágyazott CSS stílus megadása:**

A szokásos eljárás az, hogy egy webhelyen belül az összes HTML oldalhoz külső stíluslapo(ka)t kapcsolunk (lásd b/3). Egyoldalas vagy különálló weboldal esetén, ill. ha egy web-

hely egy adott oldalán a külső stíluslap(ok)hoz képest kis mértékben meg akarjuk változtatni a formázást, használatos az alábbi, belső stílust vagy beágyazott stílust definiáló <*style*> elem:

```
<style>
  CSS formázás kódolása     ( ezt lásd a CSS részben )
</style>
```

A <*style*> páros címkével tehát egy nem-HTML kódolás ágyazódik be egy HTML oldalba.

Megjegyzés: A HTML-lapokat a böngészők akkor is minden információval, logikus struktúrában megjelenítik, ha nincsen hozzájuk (sem belső, sem külső) stílus definiálva, de a prezentáció esztétikai elemei nélkül a weblap monoton, jellegtelen és színtelen lesz.

A fej HTML-kódolása tehát:

```
<head>
   <meta charset="utf-8">
   <title>…….</title>
   <link rel="stylesheet" href="……..css">
   <style>……..</style>
</head>
```

c) A **törzs** (body=törzs) megadása:

```
<body>
   ……weboldal tartalma…
</body>
```

A törzs az, amit a böngésző megjelenít, a CSS stílussal való formázás is csak erre vonatkozik. A dokumentumtípus meghatározás és a fej nem formázható, de nem is lenne értelme, hiszen azok elsősorban a böngészőknek és keresőknek szóló információkat tartalmazzák.

A weblap immár teljes kódolt szerkezete az előzőek alapján:

```
<!doctype html>
<html lang="hu">
      <head>
            <meta charset="utf-8">
            <title>……..</title>
            <link rel="stylesheet" href="……..css">
            <style>……..</style>
      </head>
      <body>
         ……weboldal tartalma……
      </body>
</html>
```

Annak nincsen jelentősége, ha a jobb áttekinthetőség érdekében a szerkezeti elemek között üres sort hagyunk (vagy nem hagyunk). Lehetne vízszintesen is egymásba ágyazni az egyes elemeket, azonban áttekinthetőségi szempontból az új sorokba írt, szerkezeti egység szerint behúzással is növelt tagolás a leghatékonyabb – bár némi fájlméret növekedést okoz.

A felhasznált betűtípusnak nincsen a kódolás szempontjából jelentősége, mi – ugyancsak az áttekinthetőség remélhető elősegítése céljából - félkövér dőlten írjuk a címkéket és jellemzőket, és normál betütípussal az értékeket ill. a szövegeket.

A weblap szerkezeti felépítéséhez a következő HTML címkéket használtuk fel:

Címke:	Funkciója:	
<!doctype>	dokumentumtípus definíció (valójában nem címke, hanem deklaráció)	páratlan címke
<html>	HTML dokumentumot definiál	páros címke
<head>	a dokumentumra vonatkozó információkat definiál	páros címke
<body>	a dokumentum törzsét definiálja	páros címke
<title>	a dokumentum címét definiálja	páros címke
<link>	a dokumentum és egy külső forrás közötti kapcsolatot definiál	páratlan címke
<meta>	a dokumentumra vonatkozó meta-adatokat definiál	páratlan címke
<style>	belső vagy beágyazott stílus definíciót hoz létre	páros címke

2.3. Az első konkrét weblap

Az alkalmazott editorral egy új HTML fájlt nyitva az előző fejezetben megadott kódolt weblap szerkezetet kell beírni, a választott témához illő címmel és egy választott CSS-stíluslap fájlnévvel. A tankönyvekben szokásosan alkalmazott kezdő cím pl. *Az első weblapom*, a stíluslapnév pl. *display.css*, a belső stílus pedig egyelőre üresen marad.

A törzsbe beírva egy ugyancsak szokásosan használt első szöveget:

Helló világ! Legyen szép napod!

(Ez azért is előnyös, mert magyar ékezetes betűket is tartalmaz, tehát a karakterkészlet ellenőrzésére is használható.)

A HTML dokumentumnak ekkor így kell kinéznie:

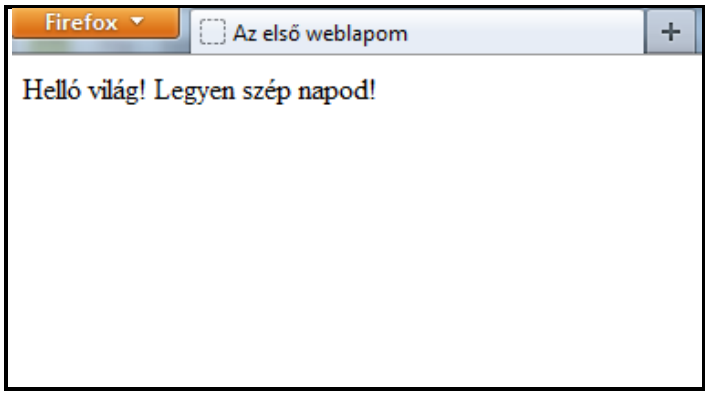
```
<!doctype html>
<html lang="hu">
    <head>
        <meta charset="utf-8">
        <title>Az első weblapom</title>
        <link rel="stylesheet" href="display.css">
        <style>
        </style>
    </head>
    <body>
        Helló világ! Legyen szép napod!
    </body>
</html>
```

(A stílusra még nincsen semmi megadva, a HTML-gyakorlás közben azonban így is érdemes beírni a CSS-re utaló címkéket.)

A HTML-lapot egy erre a célra létrehozott gyűjtőmappába kell menteni. Minden további változtatást először szintén menteni („CTR+S” vagy „Fájl/Mentés”) kell, mielőtt megnézzük az F5-el frissített eredményt, melyet a főbb böngészők mindegyikével érdemes ellenőrizni.

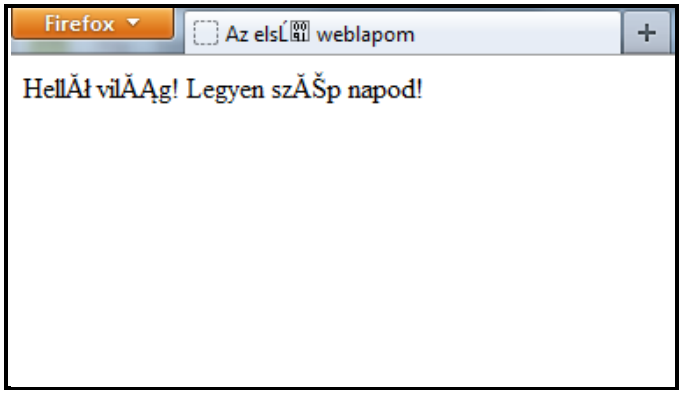
Figyelem! Ha az editor a mentést *.txt* kiterjesztéssel végzi el, a kiterjesztéseket *.html*-re ill. *.css*-re kell cserélni.

A fenti (nagyon egyszerű, de már) működőképes weboldalt így mutatják a böngészők:



Látható, hogy a HTML-kódolás fej részében megadott információból csak a címet mutatja (a címsorában) a böngésző, a törzsben létrehozott tartalom az, amit weboldalként megjelenít.

Ha a karakterkészlet hibásan van megadva, a weblapon a fenti kódolás pl. így jelenhet meg:



A *lang="hu"* jellemző/érték elhagyása - vagy más főnyelv definiálása (pl. *sk* = szlovák, *cz* = cseh, stb.) – nem befolyásolja a megjelenítést, csak a keresőprogramok keresési hatásfokát ronthatja.

2.4. Alapvető HTML címkék

Elválasztó karakterek (szóköz, tabulátor, új sor) beiktatásával változtassuk az editorral a szöveg szavai közötti távolságot, a mondatok közötti távolságot és a szöveg kezdetének helyzetét, ill. írjuk a második mondatot új sorba. Elmentve a változtatásokat és egy böngészővel megnézve az eredményt látható, hogy az editorokban használt formázást a HTML figyelmen kívül hagyja - az elválasztókarakter típusától és számától függetlenül mindig csak egy szóközt használ, folyamatosan kezeli a tartalmat, a lap bal felső sarkától a sor elején kezd, a sor végén magától vált új sorba.

A tartalom szerkezetének kialakítására és tagolására tehát különböző HTML címkéket kell használni!

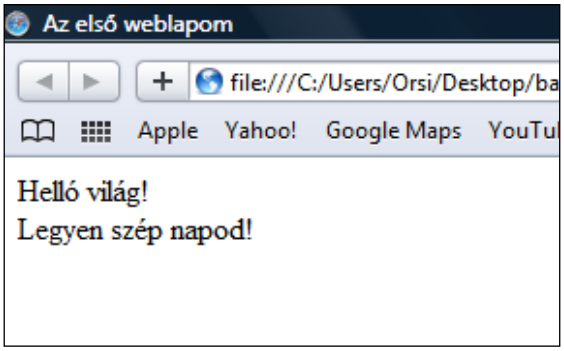
2.4.1. **Új sor** kezdését (sortörést) a <*br*> (*break* = megszakítás) páratlan címke (üres elem) definiálja. Az előző szövegünkbe beírva a sortörés címkéjét:

```
Helló világ!<br>Legyen szép napod!
```

Természetesen ez ekvivalens módon így is megadható:

```
Helló világ!<br>
Legyen szép napod!
```

A fenti HTML kódolás megjelenítése egy böngészőben:



A szövegben a második mondat így már csakugyan új sorban jelenik meg. Egy vers sorai vagy egy postacímnek a szokásosan tördelt megadása is a <*br*> címkével alakítható ki.

Figyelem! Alcímet, új bekezdést vagy függőleges felsorolást nem szabályos ilyen módon szerkeszteni, azoknak saját HTML címkéjük van. A nagyobb sorközt ugyancsak nem üres sor(ok) <*br*> címkével való beiktatásával, hanem CSS formázással kell létrehozni.

2.4.2. A **címsor**ok páros címkéi a <*h1*>……<*/h1*>-től <*h6*>……<*/h6*>-ig terjedő címsor hierarchia (h=*heading*=címsor). Ilyen módon max. 6 címet lehet egymás alá rendelni, de ennél általában lényegesen kevesebbre, max. 3-4-re van szükség.

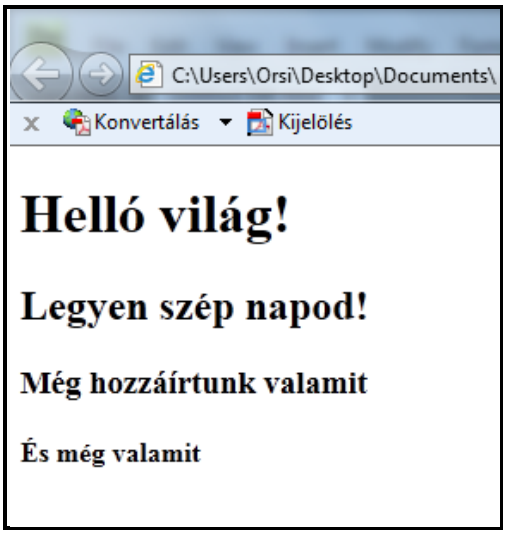
Figyelem! A *h1*….*h6* címsorok nem keverendők össze a *title* címmel, mely a weboldal elnevezését adja meg.

Ha a már meglévő szövegünket kétszintű címsorként (főcím, alcím) használjuk, és a további címsor-hierarchia bemutatására további alcímekként még hozzáírunk valamit, akkor a kódolás:

```
<h1>Helló világ!</h1>
<h2>Legyen szép napod!</h2>
<h3>Még hozzáírtunk valamit</h3>
<h4>És még valamit</h4>
```

Egy címsor a HTML szerint blokkszintű elem, azaz külön sorban kezdődik és az utána következő elem automatikusan új sorba kerül. A böngészők alapértelmezett beállítása szerint félkövér betűkkel van szedve, a betűméret a *h1*-nél a legnagyobb, majd a szám növekedésével (*h2, h3,…*) egyre kisebb.

A böngészőkkel az alapértelmezett megjelenítés:



Megjegyzés: A címsorokat mindig a tartalmi fontosságuk szerint, és soha nem a megjelenés alapján kell kódolni. Bármelyik címsorból CSS-el a fentitől eltérő, tetszőleges megjelenítés hozható létre.

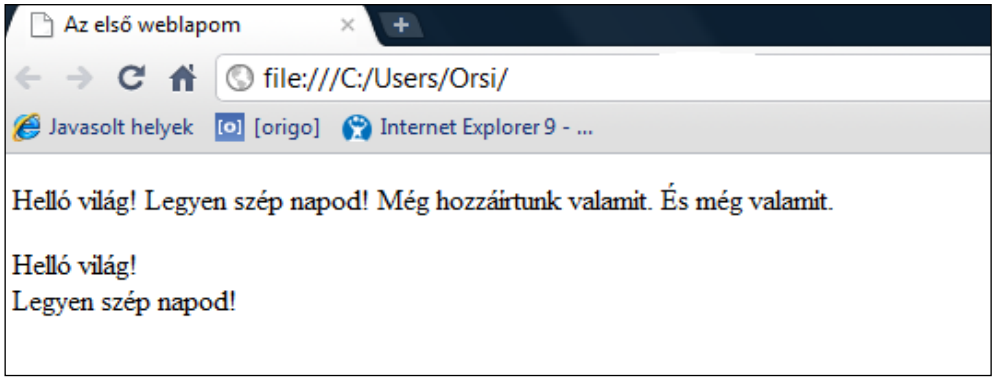
A CSS rész Betűtípusok, Színek definiálása, Blokkszintű elemek elhelyezése I. és Szöveg fejezetekben foglaltakkal formázhatók a címsorok.

2.4.3. A **bekezdés** páros címkéje a * …… * (p=*paragraph*=bekezdés). A korábban használt szövegből készítve két bekezdést:

```
<p>Helló világ! Legyen szép napod! Még hozzáírtunk valamit. És
   még valamit.</p>
<p>Helló világ!<br>Legyen szép napod!</p>
```

A böngészők a bekezdéseket is blokkszintű elemként kezelik, azaz külön sorban kezdődnek és az utánuk következő elem automatikusan új sorba kerül. Szövegbehúzás nincsen, két bekezdés között egy üres sort hagynak. A böngészők alapértelmezett beállításaitól eltérő megjelenítést CSS-el kell létrehozni.

A böngészők alapértelmezetten így jelenítik meg a bekezdéseket:



Figyelem! A második bekezdésből az is látszik, hogy a HTML címkék egymásba is ágyazhatók, lásd a * …..
….. * kódolást. Jelen esetben páratlan címke (*
*) került a *p* elembe, de ha páros címkéről lenne szó, a beágyazott részt a beágyazó rész lezárása előtt be kell zárni, különben nem egyértelmű, melyik címke melyik tartalomrészhez tartozik (rosszul beágyazott címkék = *misnested tags*).

Megjegyzés: A törzs (*body*) részbe beírt sima szövegeket mindig érdemes * *… …..…* * bekezdésekként kódolni, hogy a böngészők a formázásnál ne tévesszenek, tehát az első mondatunkat is célszerűbb lett volna

```
<p>Helló világ! Legyen szép napod!</p>
```

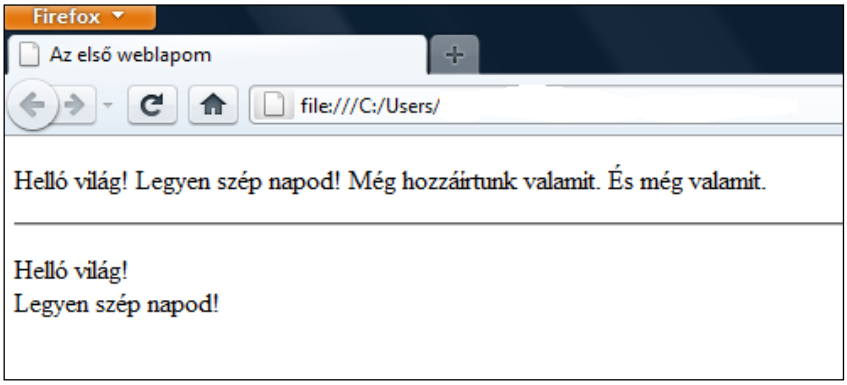
módon beírni (ha már akkor ismert lett volna a *p* címke). Általában szöveges tartalomban a bekezdés az egyik leggyakoribb elem - ha nem címsorról vagy felsorolásról van szó, akkor az esetek nagy többségében a szöveg bekezdésként jelölendő.

2.4.4. A bekezdésekkel történő tematikus tagolás egy **vízszintes vonallal** tovább fokozható, melynek kódja *<hr>* (hr=*horizontal line* =vízszintes vonal). A *<hr>* blokkszintű üres elem.

A korábbi két bekezdésünket vízszintes vonallal még jobban szétválasztva:

```
<p>Helló világ! Legyen szép napod! Még hozzáírtunk valamit. És
   még valamit.</p>
<hr>
<p>Helló világ!<br>Legyen szép napod!</p>
```

A böngészők megjelenítési eredménye:



2.4.5. A kódolás során tehetünk magunknak olyan **megjegyzés**eket/emlékeztetőket, melyek elősegíti a munkát. A megjegyzés (*comment*) páratlan címke, a *<!--* (*comment start delimiter*) és *-->* (*comment end delimiter*) által befoglalt rész a forráskódban benne marad, de a böngésző nem jeleníti meg, nem veszi figyelembe.

Figyelem! Minden weblap kódolása mindegyik böngészővel megnézhető. A megnyitott weblap egy üres területére jobb egérrel kattintva a megjelenő menüből *Forrás megtekintése* (Internet Explorer), *Oldal forrása* (Firefox), *Oldal forrásainak megtekintése* (Chrome), *Forráskód* (Opera) pontot választva a böngésző megmutatja a weboldal kódolását, beleértve a megjegyzés(eke)t is – tehát olyat ne írjunk bele, amit nem szeretnénk, ha bárki láthat.

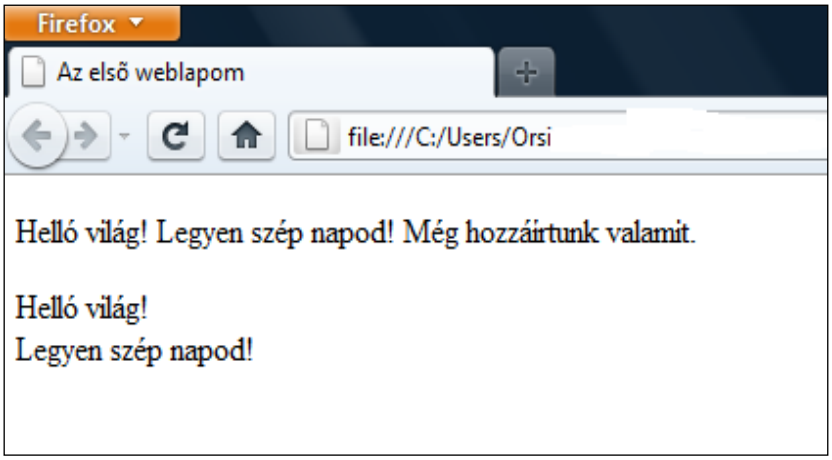
Másik alkalmazása a megjegyzésnek, ha ideiglenesen ki akarunk hagyni részeket a kódolásból. A megjegyzés címkében foglalt rész így olyan, mintha ott sem lenne, de nem kell törölni, és bármikor ismét felhasználható vagy ismét figyelmen kívül hagyható, ha a *<!--* ……..*-->* címkét kivesszük a kódolásból vagy újból beírjuk.

A megjegyzés címke fenti két alkalmazásának bemutatására

- szúrjunk be magunknak egy megjegyzést - *Ez a mondat nem fog látszani a weblapon*
- továbbá a vízszintes vonalat és az *És még valamit* mondatot ideiglenesen vegyük ki a megjelenítésből:

```
<p>Helló világ! Legyen szép napod! Még hozzáírtunk valamit.
<!--És még valamit.--></p>
<!--<hr>-->
<p>Helló világ!<br>Legyen szép napod!</p>
<!--Ez a mondat nem fog látszani a weblapon-->
```

A böngészők ezt így mutatják:



A felhasznált alapvető HTML-címkék:

Címke:		Funkciója:	
<h1>……..<h6>	</h1>……</h6>	címsort definiál	páros címke
<p>		bekezdést definiál	páros címke
 		sortörést hoz létre	páratlan címke
<hr>		vízszintes elválasztó vonalat szúr be	páratlan címke
<!--	-->	megjegyzés beszúrása vagy kódrészlet ideiglenes kiiktatása	páratlan címke

2.5. Listák

A listák nem sortörésekkel, címekkel vagy bekezdésekkel, hanem saját HTML címkékkel hozandók létre. Különös jelentőséget ad nekik, hogy a felsorolásokon kívül a weboldal menüi (amik nem mások, mint hivatkozások listái) szintén listaszerkezettel állíthatók elő.

HTML-ben három lista-fajta kódolható: számozott (rendezett), számozatlan (nem rendezett, felsorolt) és a meghatározás (definíciós, asszociációs) lista.

2.5.1. A **számozott listáknál** fontos a sorrendiség, elemei előtt felsorolásjelként számok vagy betűk állnak. A számozott lista kezdő- és zárócímkéje *
……..
* (*ordered list* = rendezett lista), az egyes listaelemeket a *
……..
* (*list item*=listaelem) páros címke definiálja.

A **számozatlan listáknál** nem fontos a sorrend, pusztán felsorolást tartalmaznak, elemei elé szimbólumok, képek is rendelhetők. A számozatlan lista kezdő- és zárócímkéje *
……* …… (*unordered list*=nem rendezett lista), az egyes listaelemeket a számozottal megegyező módon a *
……..
* (*list item*=listaelem) páros címke definiálja.

Számozott lista felépítése:

```
<ol>
      <li>.........</li>
      <li>.........</li>
            .........
            .........
            .........
</ol>
```

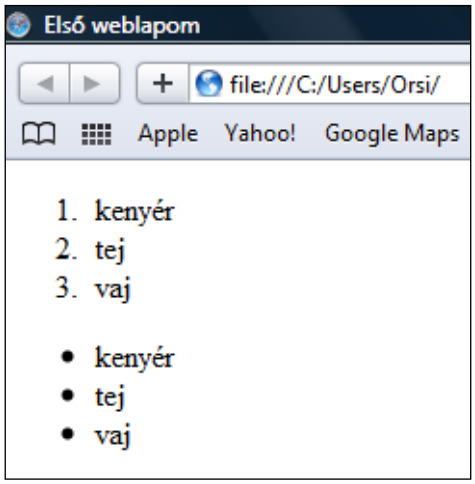
Számozatlan lista felépítése:

```
<ul>
      <li>.........</li>
      <li>.........</li>
            .........
            .........
            .........
</ul>
```

HTML fájlunkban a fenti kódolást (ismét az általánosan elterjedt) konkrét szavakkal kitöltve:

```
<ol>
      <li>kenyér</li>
      <li>tej</li>
      <li>vaj</li>
</ol>
<ul>
      <li>kenyér</li>
      <li>tej</li>
      <li>vaj</li>
</ul>
```

A böngésző alapértelmezés szerint így jeleníti meg a számozott (rendezett) és számozatlan (felsorolási) listákat:



Listák egymásba is ágyazhatók, pl. az egyes listaelemekbe újabb al-listák helyezhetők. Az így kapott, többszintűnek is nevezett listák kiválóan alkalmasak áttekinthetően tagolt tartalmak, pl. tartalomjegyzékek, munkatervek, vázlatok, stb. létrehozására.

Számozott és számozatlan listák is egymásba foglalhatók (lásd a számozatlan listában a számozott, zölddel kódolt „vajas” listát):

```
<ol>
     <li>kenyér
          <ol>
               <li>fehér kenyér</li>
               <li>barna kenyér</li>
          </ol>
     </li>
     <li>tej
          <ol>
               <li>zsíros tej</li>
               <li>zsírszegény tej</li>
          </ol>
     </li>
     <li>vaj
          <ol>
               <li>sózott vaj</li>
               <li>nem sózott vaj</li>
               <li>margarin</li>
          </ol>
     </li>
</ol>
<ul>
     <li>kenyér
          <ul>
               <li>fehér kenyér</li>
               <li>barna kenyér</li>
          </ul>
     </li>
     <li>tej
          <ul>
               <li>zsíros tej</li>
               <li>zsírszegény tej</li>
          </ul>
     </li>

     <li>vaj
          <ol>
               <li>sózott vaj</li>
               <li>nem sózott vaj</li>
               <li>margarin</li>
          </ol>
     </li>
</ul>
```

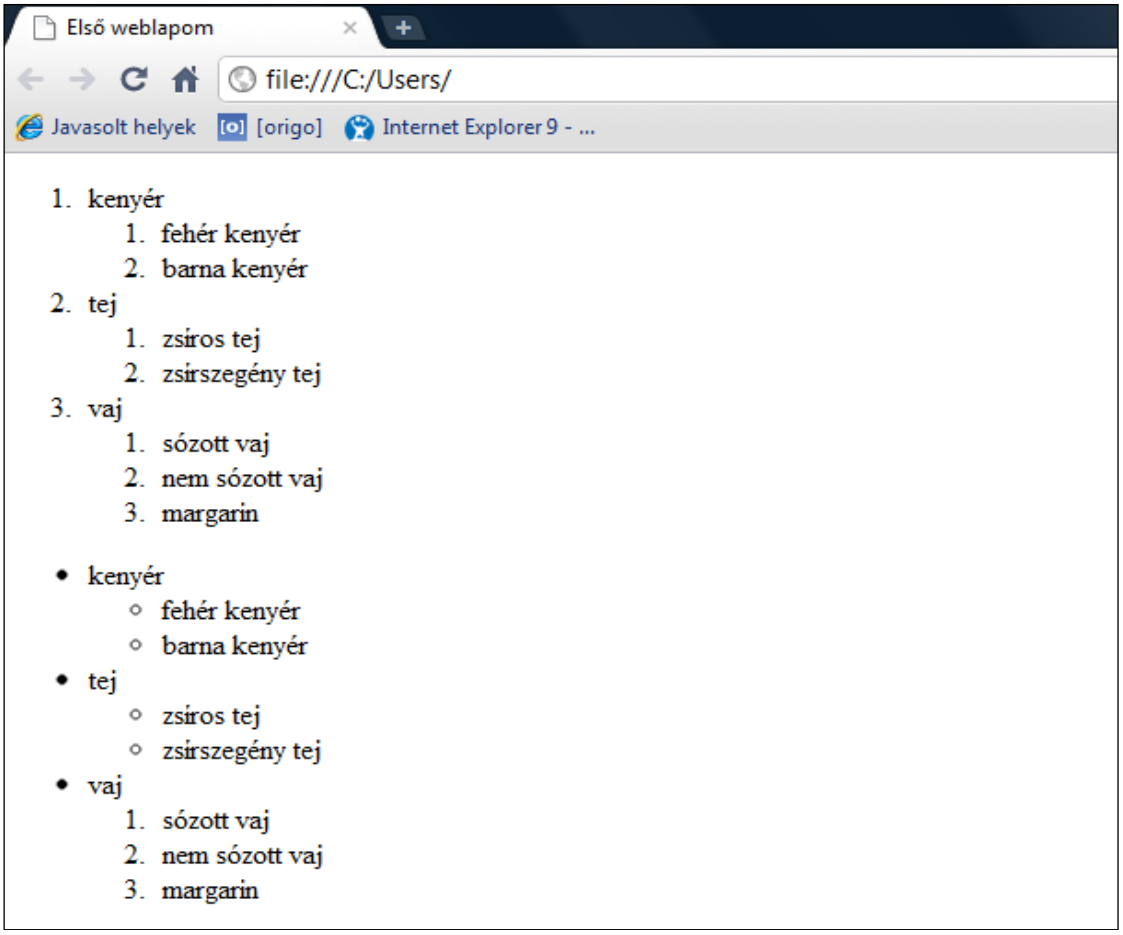
Az összetartozó kezdő- és zárócímkék felismerése érdekében célszerű az egymásba ágyazott elemeket a kódolás során behúzással (pl. a tabulátor billentyűvel) is tagolni (ennek semmi köze nincsen a weblapon való megjelenítéshez/behúzáshoz).

Figyelem! A HTML címkék korábban említett egymásba ágyazásának kérdése itt már jól megfigyelhető - a beágyazott elemeket mindig be kell zárni a beágyazó elem zárása előtt!

Megjegyzés: A HTML szabvány néhány kivételes esetben engedélyezi bizonyos páros címkéknél a zárótag elhagyását. Szándékosan nem kerülnek ezek az esetek felsorolásra, mert megtanulásuk és helyes alkalmazásuk nagyobb energiaráfordítást és hibalehetőséget jelent,

mint egyszerűen minden páros címkét zárótaggal lezárni. (Azért itt került ez a kérdés megemlítésre, mert a listák kódolásánál fordul elő az egyik ilyen kivétel – szándékosan nem is pontosítjuk, hogy az mire vonatkozik.)

A böngésző alapértelmezett megjelenítése a beágyazott lista elemeit behúzza:



2.5.2. „Rendhagyó” rendezett listák

A **számozott listák kezdőértéke** az 1-től eltérőre is beállítható, a listaelemekhez általunk **választott értékek** rendelhetők, ill. a listaelemek **csökkenő értékekkel** (visszafelé) is számozhatók.

a) Az alapértelmezetten 1-től induló növekvő számozást a számozott lista kezdő címkéjének a *start* (induló) jellemzővel és a kívánt kezdő értékkel való kiegészítésével lehet tetszőleges pozitív egész számra beállítani.

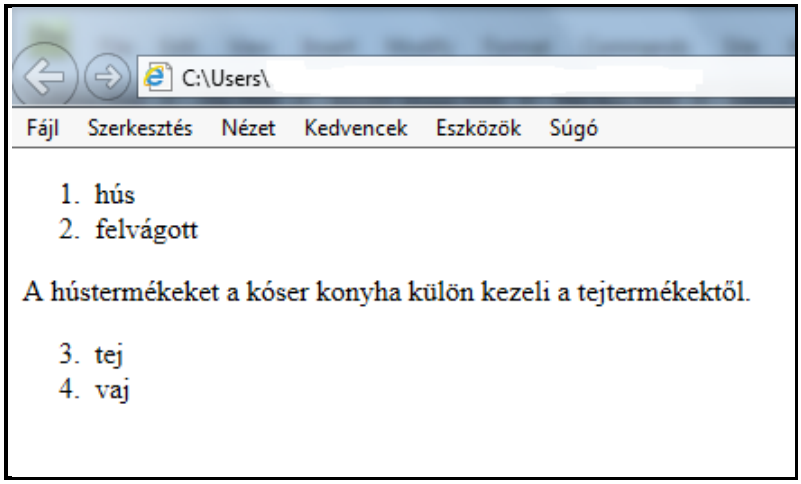
Ennek akkor lehet például jelentősége, ha egy számozott listába beszúrunk egy szöveget, majd a számozott lista folytatásának számozását nem ismét 1-től, hanem folytatólagosan kívánjuk megadni.

Ha az első számozott élelmiszerlistánk elején a kenyér helyére két új tétel (hús, felvágott) majd egy magyarázó szöveg kerül, a folytatott listát 3-ról induló számozással kódolva:

```
<ol>
    <li>hús</li>
    <li>felvágott</li>
</ol>
```

```
<p>A hústermékeket a kóser konyha külön kezeli a
   tejtermékektől.</p>
<ol start="3">
     <li>tej</li>
     <li>vaj</li>
</ol>
```

A böngészőkben való megjelenítés:

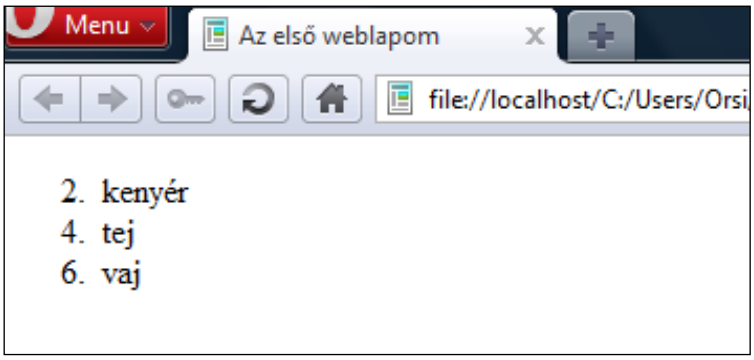


b) Az alapértelmezetten 1. 2. 3. –al haladó növekvő számozást a listaelemek kezdő címkéihez rendelt *value* (érték) jellemzővel és a kívánt értékkel lehet más sorszámozásra módosítani. Ennek akkor van például jelentősége, ha egy számozott listából kiragadunk listaelemeket, de mégis meg akarjuk őrizni az eredeti számozást.

Maradva a számozott élelmiszeres listánknál, de 2. 4. és 6.-ra módosítva a számozást:

```
<ol>
     <li value="2">kenyér</li>
     <li value="4">tej</li>
     <li value="6">vaj</li>
</ol>
```

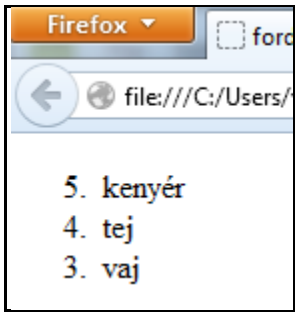
A böngészőkben a megjelenítés:



c) Az alapértelmezetten növekvő számozást a számozott lista kezdő címkéjéhez megadott *reversed* (fordított) jellemzővel lehet visszafelé számozásra módosítani. (Slágerlistákat, versenyek helyezettjeit, stb. gyakran visszafelé sorszámozva adják meg.) A *reversed* jellemző állhat érték nélkül. Az *<ol reversed>* elemet pedig a *start* jellemzővel is bővítve tetszőleges általunk választott kezdőértékről indítható a visszafelé számozás.

Az élelmiszeres listánk pl. 5-től visszafelé számozva:

```
<ol reversed start="5">
     <li>kenyér</li>
     <li>tej</li>
     <li>vaj</li>
</ol>
```



Figyelem! Az Internet Explorer-ek nem értelmezi a visszafelé számozást (5.-től felfelé növekvő számozást mutatnak helyette).

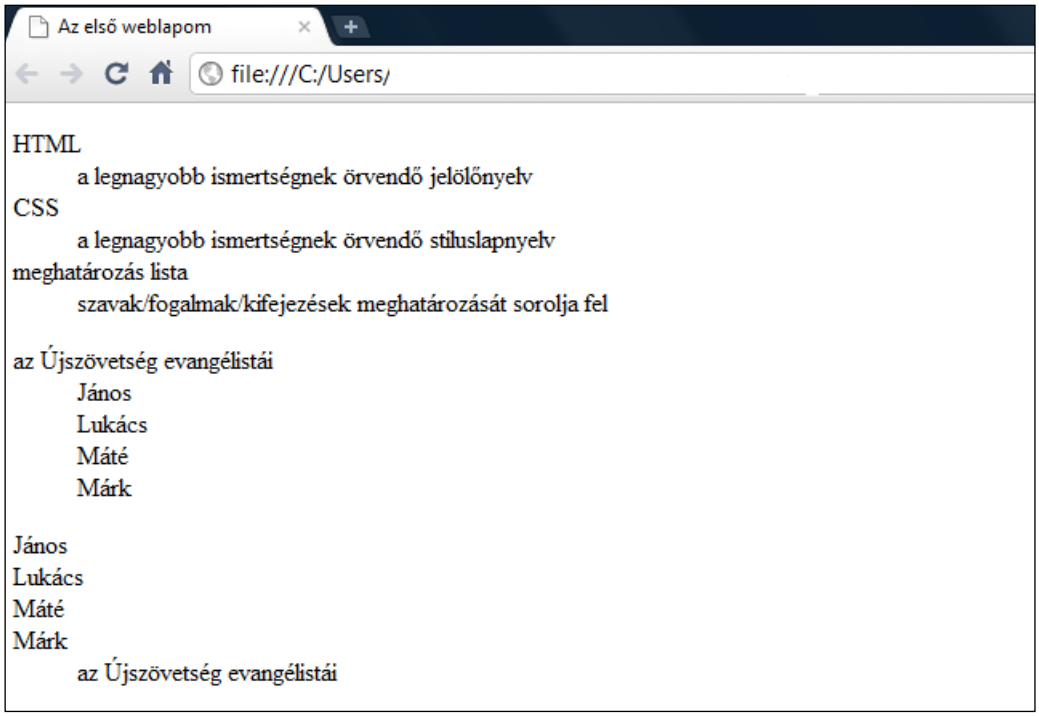
2.5.3. A **meghatározás lista** (definíciós, asszociációs lista) annyiban tér el a számozott és számozatlan listáktól, hogy nem felsorolásokat, hanem szavakat/fogalmakat/kifejezéseket és azok meghatározását társítja egymáshoz. A lista páros címkéje a *<dl>……..…..…..</dl>* (*definition list*=meghatározás lista), a szavak/fogalmak/kifejezések páros címkéje a *<dt>……..…..</dt>* (*definition term*=meghatározandó kifejezés), és magáé a meghatározásé a *<dd>……..…..</dd>* (*definition data*=meghatározási adat) páros címke. Példa:

```
<dl>
     <dt>HTML</dt>
          <dd>a legnagyobb ismertségnek örvendő jelölőnyelv</dd>
     <dt>CSS</dt>
          <dd>a legnagyobb ismertségnek örvendő stíluslapnyelv
          </dd>
     <dt>meghatározás lista</dt>
          <dd>szavak/fogalmak/kifejezések meghatározását sorolja
          fel</dd>
</dl>
```

Egy szóhoz több meghatározás, ill. több szóhoz egy meghatározás is hozzárendelhető:

```
<dl>
     <dt>az Újszövetség evangélistái</dt>
          <dd>János</dd>
          <dd>Lukács</dd>
          <dd>Máté</dd>
          <dd>Márk</dd>
</dl>
<dl>
     <dt>János</dt>
     <dt>Lukács</dt>
     <dt>Máté</dt>
     <dt>Márk</dt>
          <dd>az Újszövetség evangélistái</dd>
</dl>
```

Alapértelmezetten behúzott *dd* – címkével jelenítik meg a definíciós listákat a böngészők:



Meghatározás listával kódolhatók pl. a szótárak/szószedetek, játékszabályok felsorolása, gyakran feltett kérdések (GYIK) és a rájuk adott válaszok, stb.

A listák alapértelmezettől eltérő formázása – beleértve a vízszintes kialakítást is - CSS-el történik. *Lásd a CSS rész Listák formázása és a Blokkszintű elemek elhelyezése I.-II. fejezeteket.*

Valamennyi lista blokkszintű elem, azaz külön sorban kezdődik, és az utána következő elem automatikusan új sorba kerül.

<u>A listáknál felhasznált HTML címkék:</u>

Címke:	Funkciója:	
	számozott (rendezett) listát definiál	páros címke
	számozatlan (felsorolási) listát definiál	páros címke
	lista elemet definiál	páros címke
<dl>	meghatározás (definíciós) listát definiál	páros címke
<dt>	meghatározandó szót/fogalmat/kifejezést definiál	páros címke
<dd>	meghatározást/társítást definiálja	páros címke

2.6. Táblázatok

Egy táblázat feladata az összetartozó adatok egymáshoz való viszonyának áttekinthető módon való megjelenítése. A táblázat sorokból, oszlopokból, fejlécekből és (opcionálisan) a táblázat címéből áll. A sorok és oszlopok által alkotott cellák tartalmazzák a táblázat adatait, mely nem csak szöveg és szám, hanem akár újabb táblázat, lista vagy kép is lehet. A táblázat alkotórészeinek külön címkéjük van, tehát a táblázat címét nem a *<h1.....</h1>* címsor címkével, sorait nem a * .... * bekezdés címkékkel vagy a *
* sortöréssel lehet megadni.

2.6.1. A táblázat (*table*=táblázat) nyitó- és zárócímkéje a *<table>……</table>*. A vízszintes sorokat egyenként a *<tr>……</tr>* páros címke (*table row*=táblázat sor), a cellákban lévő adatokat egyenként a *<td>…..</td>* páros címke *(table data*=táblázat adat), a sorok és oszlopok fejléceit egyenként a *<th>……</th>* páros címke (*table header*=táblázat fejléc), a címet pedig a *<caption>……</caption>* páros címke (*caption*=cím) definiálja.

Egy HTML-ben kódolt táblázat kialakításának általános szabályai:

- a táblázat minden oszlopában azonos számú cellának kell lennie
- egy adott oszlopban a cellák szélessége mindig azonos
- az azonos számú és szélességű celláktól csak a következőkben tárgyalt sor- és oszlop-átfogással (azaz cellák egyesítésével) lehet egész cellányi lépésekben eltérni
- amennyiben a táblázat üres cellát is tartalmaz, azt is be kell kódolni *<td></td>* vagy *<th></th>* üres tartalommal (egyébként borul a táblázat szerkezete)

Ha már adatokkal kell dolgozni, válasszunk webszerkesztőnek hasznosakat – kódoljuk *Böngészők statisztikája 2013. szeptember - december* címmel a magyar weblapok magyar látogatottságát (letöltések számát) böngészőcsoportok szerinti megoszlásban bemutató alábbi táblázatot:

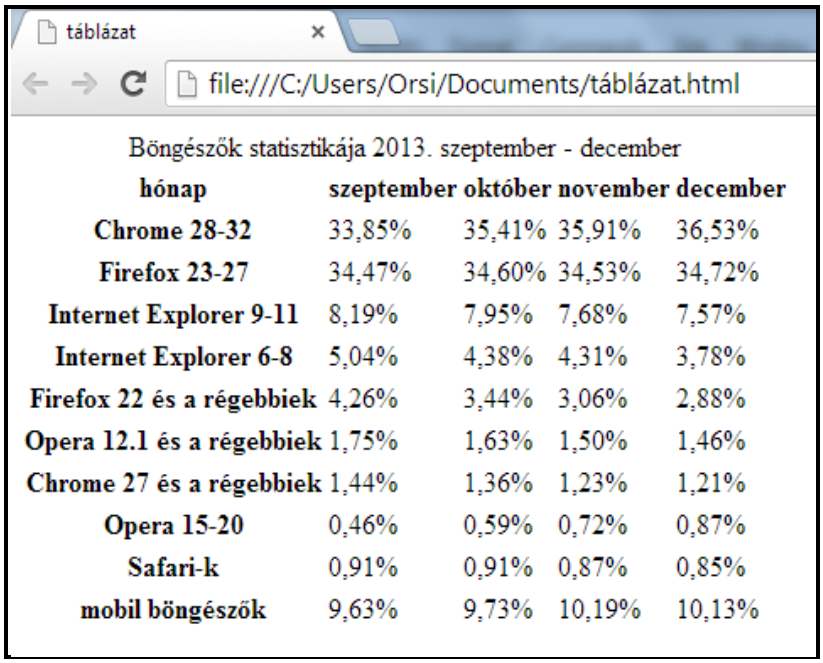
Böngészők statisztikája 2013. szeptember - december				
hónap	szeptember	október	november	december
Chrome 28-32	33,85%	35,41%	35,91%	36,53%
Firefox 23-27	34,47%	34,60%	34,53%	34,72%
Internet Explorer 9-11	8,19%	7,95%	7,68%	7,57%
Internet Explorer 6-8	5,04%	4,38%	4,31%	3,78%
Firefox 22 és a régebbiek	4,26%	3,44%	3,06%	2,88%
Opera 12.1 és a régebbiek	1,75%	1,63%	1,50%	1,46%
Chrome 27 és a régebbiek	1,44%	1,36%	1,23%	1,21%
Opera 15-20	0,46%	0,59%	0,72%	0,87%
Safari-k	0,91%	0,91%	0,87%	0,85%
mobil böngészők*	9,63%	9,73%	10,19%	10,13%

Megjegyzés: A döntő részt az Android/Chrome Mobile-ok, ill. a Safari Mobile-ok képviselik, az IE Mobile és Opera Mini csak 0,5%-0,5%-os, az Opera Mobile és Nokia Browser kb. 0,15%-0,15%-os részesedéssel rendelkeznek, a Firefox for Android és Blackberry Browser a mérhetőség határán vannak.

A táblázat kódolása a *<table>……</table>* címkék között helyezkedik el. Logikailag első a cím - bár ha a sorok közé vagy a táblázat végére kódoljuk, akkor is a böngészők alaplyos alakú a táblázat, nincsenek üres és/vagy összevonandó cellák:

```
<table>
      <caption>Böngészők statisztikája 2013. szeptember -
               december</caption>
      <tr>
            <th>hónap</th><th>szeptember</th><th>október</th>
            <th>november</th><th>december</th>
      </tr>
      <tr>
            <th>Chrome 28-32</th><td>33,85%</td><td>35,41%</td>
            <td>35,91%</td><td>36,53%</td>
      </tr>
      <tr>
            <th>Firefox 23-27</th><td>34,47%</td><td>34,60%</td>
            <td>34,53%</td><td>34,72%</td>
      </tr>
      <tr>
            <th>Internet Explorer 9-11</th><td>8,19%</td>
            <td>7,95%</td><td>7,68%</td><td>7,57%</td>
      </tr>
      <tr>
            <th>Internet Explorer 6-8</th><td>5,04%</td>
            <td>4,38%</td><td>4,31%</td><td>3,78%</td>
      </tr>
      <tr>
           <th>Firefox 22 és a régebbiek</th><td>4,26%</td>
           <td>3,44%</td><td>3,06%</td><td>2,88%</td>
      </tr>
      <tr>
            <th>Opera 12.1 és a régebbiek</th><td>1,75%</td>
            <td>1,63%</td><td>1,50%</td><td>1,46%</td>
      </tr>
      <tr>
            <th>Chrome 27 és a régebbiek</th><td>1,44%</td>
            <td>1,36%</td><td>1,23%</td><td>1,21%</td>
      </tr>
      <tr>
            <th>Opera 15-20</th><td>0,46%</td><td>0,59%</td>
            <td>0,72%</td><td>0,87%</td>
      </tr>
      <tr>
            <th>Safari-k</th><td>0,91%</td><td>0,91%</td>
            <td>0,87%</td><td>0,85%</td>
      </tr>
      <tr>
            <th>mobil böngészők</th><td>9,63%</td><td>9,73%</td>
            <td>10,19%</td><td>10,13%</td>
      </tr>
</table>
```

Alapértelmezetten a táblázat keret nélkül, a cellatartalomnak megfelelő oszlopszélességgel és sormagassággal jelenik meg, a fejlécek (*th* elemek) középre igazítva félkövéren, az adatok (*td* elemek) balra zárva normál betűtípussal formázottak:



Egy-egy cella több oszlopot vagy több sort is átfoghat. A *th* (fejléc) és *td* (adat) páros címkékhez megadott **colspan** (oszlop átfogás), ill. **rowspan** (sor átfogás) jellemzőkkel és az átfogandó oszlopok ill. sorok számának, mint értéknek a megadásával kódolható egy-egy cellaegyesítés (*spanning* = átívelés).

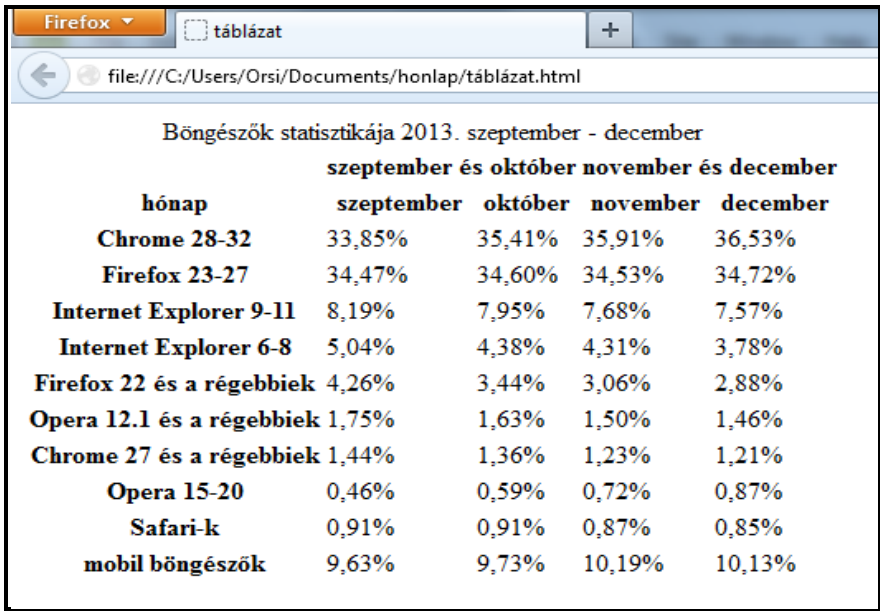
Az oszlopátfogás bemutatására a 2013. szeptember - decemberi statisztikát úgy adjuk meg, hogy egy üres cellával kezdődő új sort is adunk a táblázathoz, melyben a kéthavonkénti időintervallumot fel tudjuk tüntetni, így a két új szöveges cella oszlopátfogásával az alábbi táblázat adódik:

Böngészők statisztikája 2013. szeptember - december				
	szeptember és október		november és december	
hónap	szeptember	október	november	december
Chrome 28-32	33,85%	35,41%	35,91%	36,53%
Firefox 23-27	34,47%	34,60%	34,53%	34,72%
Internet Explorer 9-11	8,19%	7,95%	7,68%	7,57%
Internet Explorer 6-8	5,04%	4,38%	4,31%	3,78%
Firefox 22 és a régebbiek	4,26%	3,44%	3,06%	2,88%
Opera 12.1 és a régebbiek	1,75%	1,63%	1,50%	1,46%
Chrome 27 és a régebbiek	1,44%	1,36%	1,23%	1,21%
Opera 15-20	0,46%	0,59%	0,72%	0,87%
Safari-k	0,91%	0,91%	0,87%	0,85%
mobil böngészők	9,63%	9,73%	10,19%	10,13%

Az oszlopátfogásokat és az üres cellát is tartalmazó táblázat kódolása:

```
<table>
     <caption>Böngészők statisztikája 2013. szeptember -
              december</caption>
     <tr>
          <th></th><th colspan="2">szeptember és október</th>
          <th colspan="2">november és december</th>
     </tr>
     <tr>
          <th>hónap</th><th>szeptember</th><th>október</th>
          <th>november</th><th>december</th>
     </tr>
     <tr>
          <th>Chrome 28-32</th><td>33,85%</td><td>35,41%</td>
          <td>35,91%</td><td>36,53%</td>
     </tr>
     <tr>
          <th>Firefox 23-27</th><td>34,47%</td><td>34,60%</td>
          <td>34,53%</td><td>34,72%</td>
     </tr>
     <tr>
          <th>Internet Explorer 9-11</th><td>8,19%</td>
          <td>7,95%</td><td>7,68%</td><td>7,57%</td>
     </tr>
     <tr>
          <th>Internet Explorer 6-8</th><td>5,04%</td>
          <td>4,38%</td><td>4,31%</td><td>3,78%</td>
     </tr>
     <tr>
          <th>Firefox 22 és a régebbiek</th><td>4,26%</td>
          <td>3,44%</td><td>3,06%</td><td>2,88%</td>
     </tr>
     <tr>
          <th>Opera 12.1 és a régebbiek</th><td>1,75%</td>
          <td>1,63%</td><td>1,50%</td><td>1,46%</td>
     </tr>
     <tr>
          <th>Chrome 27 és a régebbiek</th><td>1,44%</td>
          <td>1,36%</td><td>1,23%</td><td>1,21%</td>
     </tr>
     <tr>
          <th>Opera 15-20</th><td>0,46%</td><td>0,59%</td>
          <td>0,72%</td><td>0,87%</td>
     </tr>
     <tr>
          <th>Safari-k</th><td>0,91%</td><td>0,91%</td>
          <td>0,87%</td><td>0,85%</td>
     </tr>
     <tr>
          <th>mobil böngészők</th><td>9,63%</td><td>9,73%</td>
          <td>10,19%</td><td>10,13%</td>
     </tr>
</table>
```

Az alapértelmezett megjelenítés:



A **sorátfogás** bemutatásával folytatva a kódolást, a fenti táblázat második és harmadik sorának egyesített kezdő cellájába beírjuk az „időintervallum” szót:

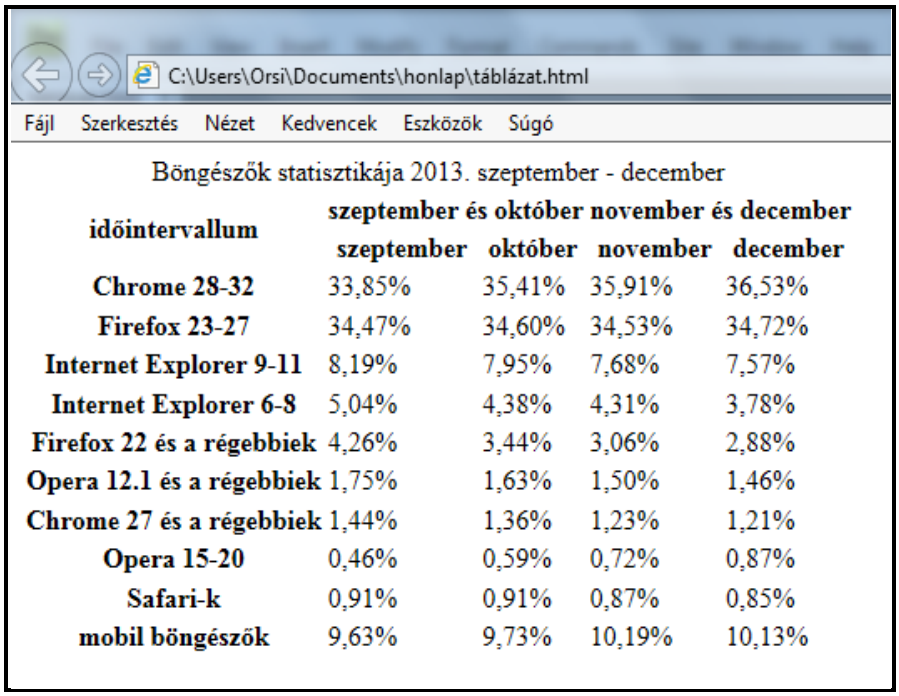
Böngészők statisztikája 2013. szeptember - december				
időintervallum	szeptember és október		november és december	
	szeptember	október	november	december
Chrome 28-32	33,85%	35,41%	35,91%	36,53%
Firefox 23-27	34,47%	34,60%	34,53%	34,72%
Internet Explorer 9-11	8,19%	7,95%	7,68%	7,57%
Internet Explorer 6-8	5,04%	4,38%	4,31%	3,78%
Firefox 22 és a régebbiek	4,26%	3,44%	3,06%	2,88%
Opera 12.1 és a régebbiek	1,75%	1,63%	1,50%	1,46%
Chrome 27 és a régebbiek	1,44%	1,36%	1,23%	1,21%
Opera 15-20	0,46%	0,59%	0,72%	0,87%
Safari-k	0,91%	0,91%	0,87%	0,85%
mobil böngészők	9,63%	9,73%	10,19%	10,13%

A módosított táblázat kódolásakor a második és harmadik sor kezdő celláit (*üres* ill. *hónap*) ki kell hagyni, hiszen ezek helyét foglalja el a sorátfogásos (*időintervallum*) cella:

```
<table>
    <caption>Böngészők statisztikája 2013. szeptember -
            december</caption>
    <tr>
        <th rowspan="2">időintervallum</th><th colspan="2">
        szeptember és október</th><th colspan="2">november
        és december</th>
    </tr>
```

```
    <tr>
        <th>szeptember</th><th>október</th><th>november</th>
        <th>december</th>
    </tr>
    <tr>
        <th>Chrome 28-32</th><td>33,85%</td><td>35,41%</td>
        <td>35,91%</td><td>36,53%</td>
    </tr>
    <tr>
        <th>Firefox 23-27</th><td>34,47%</td><td>34,60%</td>
        <td>34,53%</td><td>34,72%</td>
    </tr>
    <tr>
        <th>Internet Explorer 9-11</th><td>8,19%</td>
        <td>7,95%</td><td>7,68%</td><td>7,57%</td>
    </tr>
    <tr>
        <th>Internet Explorer 6-8</th><td>5,04%</td>
        <td>4,38%</td><td>4,31%</td><td>3,78%</td>
    </tr>
    <tr>
        <th>Firefox 22 és a régebbiek</th><td>4,26%</td>
        <td>3,44%</td><td>3,06%</td><td>2,88%</td>
    </tr>
    <tr>
        <th>Opera 12.1 és a régebbiek</th><td>1,75%</td>
        <td>1,63%</td><td>1,50%</td><td>1,46%</td>
    </tr>
    <tr>
        <th>Chrome 27 és a régebbiek</th><td>1,44%</td>
        <td>1,36%</td><td>1,23%</td><td>1,21%</td>
    </tr>
    <tr>
        <th>Opera 15-20</th><td>0,46%</td><td>0,59%</td>
        <td>0,72%</td><td>0,87%</td>
    </tr>
    <tr>
        <th>Safari-k</th><td>0,91%</td><td>0,91%</td>
        <td>0,87%</td><td>0,85%</td>
    </tr>
    <tr>
        <th>mobil böngészők</th><td>9,63%</td><td>9,73%</td>
        <td>10,19%</td><td>10,13%</td>
    </tr>
</table>
```

A fenti kódolás alapértelmezett megjelenítése:



Böngészők statisztikája 2013. szeptember - december

időintervallum	szeptember és október		november és december	
	szeptember	október	november	december
Chrome 28-32	33,85%	35,41%	35,91%	36,53%
Firefox 23-27	34,47%	34,60%	34,53%	34,72%
Internet Explorer 9-11	8,19%	7,95%	7,68%	7,57%
Internet Explorer 6-8	5,04%	4,38%	4,31%	3,78%
Firefox 22 és a régebbiek	4,26%	3,44%	3,06%	2,88%
Opera 12.1 és a régebbiek	1,75%	1,63%	1,50%	1,46%
Chrome 27 és a régebbiek	1,44%	1,36%	1,23%	1,21%
Opera 15-20	0,46%	0,59%	0,72%	0,87%
Safari-k	0,91%	0,91%	0,87%	0,85%
mobil böngészők	9,63%	9,73%	10,19%	10,13%

Megjegyzés: A sor- és oszlopösszevonás természetesen a *<th>……....</th>* fejlécekkel analóg módon bármelyik *<td>……...…</td>* adatcellával is kódolható. Adatcellák oszlopösszevonásánál – az alapértelmezetten balra zárt megjelenítés miatt – ügyelni kell arra, hogy ha nem egyértelmű az adat hovatartozása, akkor CSS-el középre kell azt pozícionálni.

Egy **táblázat celláinak tartalma lehet újabb táblázat vagy lista** is. Megjegyzendő azonban, hogy a táblázatok többszörös egymásba ágyazása az áttekinthetőség rovására megy és lassítja a letöltődést és megjelenítést, tehát a gyakorlatban korlátozottan célszerű a használata.

A táblázatba való újabb táblázat és lista beágyazására példaként a korábbi statisztika decemberi adatainak további részletezését használjuk fel. Az Internet Explorer 6-8 számozott, az Internet Explorer 9-11 számozatlan lista, a Safari-k táblázat formájában kerülnek bekódolásra a már meglévő, legelső táblázatunkba. Az adatok az alábbiak:

IE 6: 0,03%
IE 7: 0,14%
IE 8: 3,61%
IE 9: 1,83%
IE 10: 2,05%
IE 11: 3,69%

Safari 5.0 és régebbiek: 0,11%
Safari 5.1: 0,16%
Safari 6.0: 0,09%
Safari 6.1: 0,15%
Safari 7.0: 0,31%

A beágyazandó IE 6-8 lista:

```
<ol>
   <li>IE6:0,03%</li>
   <li>IE7:0,14%</li>
   <li>IE8:3,61%</li>
</ol>
```

A beágyazandó IE 9-11 lista:

```
<ul>
  <li>IE9:1,83%</li>
  <li>IE10:2,05%</li>
  <li>IE11:3,69%</li>
</ul>
```

A beágyazandó Safari táblázat:

```
<table>
    <tr>
        <th>Safari 5.0 és régebbiek</th><td>0,11%</td>
    </tr>
    <tr>
        <th>Safari 5.1</th><td>0,16%</td>
    </tr>
    <tr>
        <th>Safari 6.0</th><td>0,09%</td>
    </tr>
    <tr>
        <th>Safari 6.1</th><td>0,15%</td>
    </tr>
    <tr>
        <th>Safari 7.0</th><td>0,31%</td>
    </tr>
</table>
```

Nem maradt más hátra, mint a beágyazandó listák és a táblázat kódjainak beírása a már meglévő, eredeti táblázatkód vonatkozó adatcelláiba (a cellaértékek helyére):

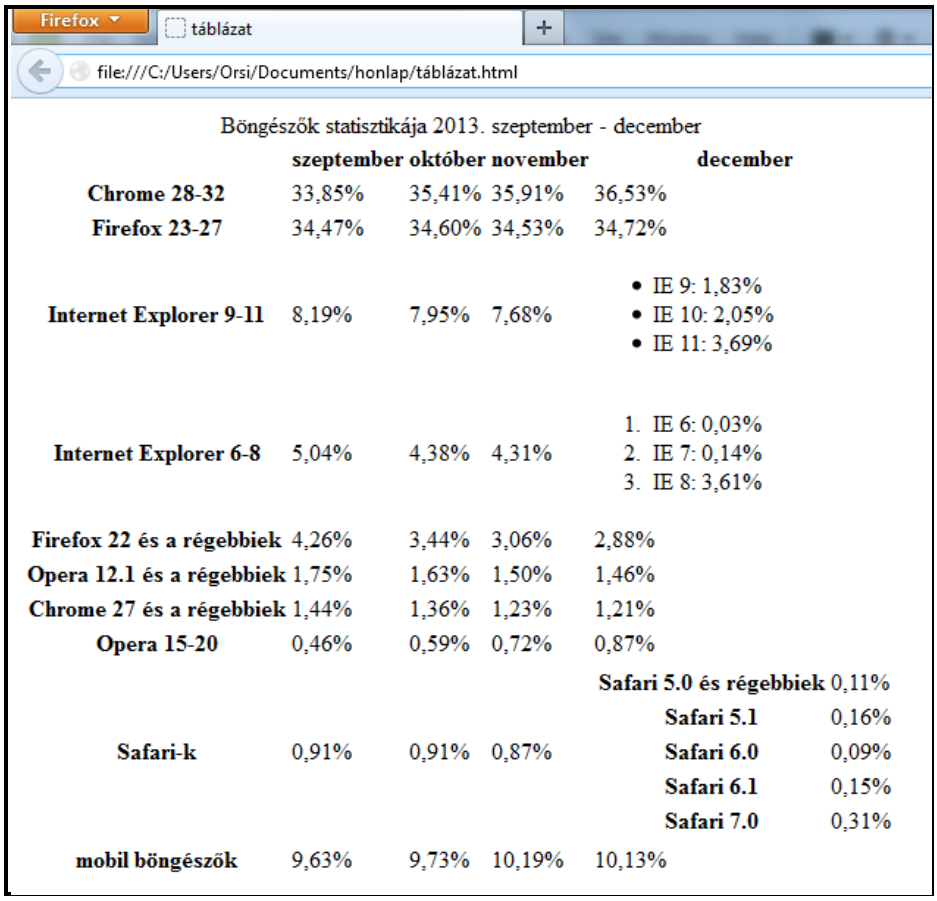
```
<table>
      <caption>Böngészők statisztikája 2013. szeptember -
                december</caption>
      <tr>
            <th></th><th>szeptember</th><th>október</th>
            <th>november</th><th>december</th>
      </tr>
      <tr>
            <th>Chrome 28-32</th><td>33,85%</td><td>35,41%</td>
            <td>35,91%</td><td>36,53%</td>
      </tr>
      <tr>
            <th>Firefox 23-27</th><td>34,47%</td><td>34,60%</td>
            <td>34,53%</td><td>34,72%</td>
      </tr>
      <tr>
            <th>Internet Explorer 9-11</th><td>8,19%</td>
            <td>7,95%</td><td>7,68%</td>
            <td><ul>
                    <li>IE9:1,83%</li>
                    <li>IE10:2,05%</li>
                    <li>IE11:3,69%</li>
                </ul></td>
      </tr>
      <tr>
            <th>Internet Explorer 6-8</th><td>5,04%</td>
            <td>4,38%</td><td>4,31%</td>
            <td><ol>
                    <li>IE6:0,03%</li>
                    <li>IE7:0,14%</li>
                    <li>IE8:3,61%</li>
                </ol></td>
      </tr>
      <tr>
          <th>Firefox 22 és a régebbiek</th><td>4,26%</td>
          <td>3,44%</td><td>3,06%</td><td>2,88%</td>
      </tr>
```

```
    <tr>
        <th>Opera 12.1 és a régebbiek</th><td>1,75%</td>
        <td>1,63%</td><td>1,50%</td><td>1,46%</td>
    </tr>
    <tr>
        <th>Chrome 27 és a régebbiek</th><td>1,44%</td>
        <td>1,36%</td><td>1,23%</td><td>1,21%</td>
    </tr>
    <tr>
        <th>Opera 15-20</th><td>0,46%</td><td>0,59%</td>
        <td>0,72%</td><td>0,87%</td>
    </tr>
    <tr>
        <th>Safari-k</th><td>0,91%</td><td>0,91%</td>
        <td>0,87%</td>
        <td><table>
            <tr>
                <th>Safari 5.0 és régebbiek</th><td>0,11%
                </td>
            </tr>
            <tr>
                <th>Safari 5.1</th><td>0,16%</td>
            </tr>
            <tr>
                <th>Safari 6.0</th><td>0,09%</td>
            </tr>
            <tr>
                <th>Safari 6.1</th><td>0,15%</td>
            </tr>
            <tr>
                <th>Safari 7.0</th><td>0,31%</td>
            </tr>
        </table></td>
    </tr>
    <tr>
        <th>mobil böngészők</th><td>9,63%</td><td>9,73%</td>
        <td>10,19%</td><td>10,13%</td>
    </tr>
</table>
```

Folyamatosan táblázatsorokba tömörítve jóval rövidebbnek tűnne a kódolás, de az áttekinthetőség kedvéért maradt a tagolt felépítés.

Mint az alábbi képen ismét látható, alapértelmezetten a fejlécek és adatcellák tartalmának megfelelő oszlopszélességeket és sormagasságokat alakítanak ki a böngészők, az adatokat balra igazítják normál karakterekkel, a fejléceket pedig középre igazítják félkövér karakterekkel. Beágyazott táblázatok/listák esetében így elég vegyes kép alakul ki (hiszen a beágyazott táblázat fejléce a befogadó táblázat adata, az egyszerű adat és a lista/táblázat a balra igazítás ellenére sem kerül feltétlenül egymás alá az egyes oszlopokban, stb.), ezért az ilyen összetett táblázat járulékos CSS formázása az áttekinthetőség érdekében szinte mindig szükséges.

Az alapértelmezett megjelenítés:



2.6.2. Nagyméretű táblázatoknál a szerkezet áttekinthetőségét segítik elő a *thead, tbody, tfoot* páros HTML-címkék. Több részre bontott táblázattörzs esetén a darabok így külön-külön is kezelhetők, hosszú táblázatoknál nyomtatáskor a táblázatfej és táblázat lábléc minden oldal tetején és alján megismételhető, és a táblázat szerkezeti egységei szerinti formázást is a fenti címkék teszik lehetővé.

thead=table head=táblázatfej (nem a táblázat cím!)
tbody=table body=táblázattörzs
tfoot=table footer=táblázat lábléce

Táblázatfej (*thead* elem) és táblázat lábléc (*tfoot* elem) egy táblázaton belül csak egy-egy lehet, de mindegyikük tartalmazhat egynél több *tr* sort is. A táblázattörzsek (*tbody* elemek) száma nincsen korlátozva.

Egy táblázatnak kódolással történő szerkezeti egységekre bontását a 2014. nyarán mért hazai böngészőhasználati statisztikán mutatjuk be:

Böngészők statisztikája 2014. nyarán			
hónap	június	július	augusztus
Chrome 30 és újabbak	37,49%	36,61%	36,58%
Firefox 25 és újabbak	32,46%	31,49%	30,94%
IE 10-11	6,72%	7,02%	7,72%

IE 8	2,37%	2,26%	2,31%
Firefox 24 és régebbiek	2,27%	1,99%	1,85%
IE 9	1,51%	1,46%	1,42%
Opera 20 és újabbak	1,33%	1,36%	0,96%
Chrome 29 és régebbiek	1,24%	1,13%	0,97%
Opera 19 és régebbiek	1,02%	0,91%	0,72%
Safari 6.0 és újabbak	0,62%	0,63%	0,48%
Safari 5.1 és régebbiek	0,27%	0,26%	0,41%
IE 6-7	0,13%	0,12%	0,20%
mobil böngészők*	12,57%	14,76%	15,44%

* ebből az Opera Mini-k kb. 0,5-0,6%

Bekódolva a táblázatot úgy, hogy a *thead*, *tbody* és *tfoot* címkéket elhelyezzük a megfelelő szerkezeti egységeknél (a táblázattörzset nem bontottuk fel több részre, tehát csak egy *tbody*-t alkalmazva):

```
<table>
    <caption>Böngészők statisztikája 2014. nyarán</caption>
    <thead>
        <tr>
            <th>hónap</th><th>június</th><th>július</th>
            <th>augusztus</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th>Chrome 30 és újabbak</th><td>37,49%</td>
                <td>36,61%</td><td>36,58%</td>
        </tr>
        <tr>
            <th>Firefox 25 és újabbak</th><td>32,46%</td>
                <td>31,49%</td><td>30,94%</td>
        </tr>
        <tr>
            <th>IE 10-11</th><td>6,72%</td>
                <td>7,02%</td><td>7,72%</td>
        </tr>
        <tr>
            <th>IE 8</th><td>2,37%</td>
                <td>2,26%</td><td>2,31%</td>
        </tr>
        <tr>
            <th>Firefox 24 és régebbiek</th><td>2,27%</td>
                <td>1,99%</td><td>1,85%</td>
        </tr>
```

```
            <tr>
                <th>IE 9</th><td>1,51%</td><td>1,46%</td>
                    <td>1,42%</td>
            </tr>
            <tr>
                <th>Opera 20 és újabbak</th><td>1,33%</td>
                    <td>1,36%</td><td>0,96%</td>
            </tr>
            <tr>
                <th>Chrome 29 és régebbiek</th><td>1,24%</td>
                    <td>1,13%</td><td>0,97%</td>
            </tr>
            <tr>
                <th>Opera 19 és régebbiek</th><td>1,02%</td>
                    <td>0,91%</td><td>0,72%</td>
            </tr>
            <tr>
                <th>Safari 6.0 és újabbak</th><td>0,62%</td>
                    <td>0,63%</td><td>0,48%</td>
            </tr>
            <tr>
                <th>Safari 5.1 és régebbiek</th><td>0,27%</td>
                    <td>0,26%</td><td>0,41%</td>
            </tr>
            <tr>
                <th>IE 6-7</th><td>0,13%</td>
                    <td>0,12%</td><td>0,20%</td>
            </tr>
            <tr>
                <th>mobil böngészők*</th><td>12,57%</td>
                    <td>14,76%</td><td>15,44%</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <td colspan="4">* ebből az Opera Mini-k kb. 0,5-
                                0,6%</td>
            </tr>
        </tfoot>
</table>
```

Megjegyzések: A táblázat lábléce általában folyamatos szöveg, melynek nem kell követnie a táblázat oszlopainak tagolását. Ilyen esetben a szöveg a *th*-ba, vagy a *th* elhagyásával az első *td* elembe írható (*th* használatakor alapértelmezetten vastag dőlt betűkkel jelenne meg a szöveg). Az oszlopátfogást azért kódoltuk be, hogy a táblázat oszlopainak szélessége ne a lábléc szövegéhez, hanem a táblázat érdemi részéhez - a fejlécekhez és adatcellákhoz – igazodjon.

Figyelem! A *tfoot* elemet kódolhattuk volna a *caption* elem alá vagy fölé, vagy a *thead* és *tbody* elemek közé, a böngészők akkor is a táblázat alján jelenítik meg a láblécet. A *thead* elemre ugyanígy érvényes ez a megállapítás – ha a táblázat végére kódoljuk, akkor is felül jelenik meg. Célszerű azonban a logikai sorrend szerint kódolni őket.

A táblázat szerkezetére vonatkozó csoportosító HTML címkék beiktatása (eltekintve a lábléc hozzáadásától) önmagában nem okozott semmilyen változást a böngészők alapértelmezett megjelenítésében:

C:\Users\Orsi\Documents\honlap\új táblázat.html

Fájl Szerkesztés Nézet Kedvencek Eszközök Súgó

Böngészok statisztikája 2014. nyarán

hónap	június	július	augusztus
Chrome 30 és újabbak	37,49%	36,61%	36,58%
Firefox 25 és újabbak	32,46%	31,49%	30,94%
IE 10-11	6,72%	7,02%	7,72%
IE 8	2,37%	2,26%	2,31%
Firefox 24 és régebbiek	2,27%	1,99%	1,85%
IE 9	1,51%	1,46%	1,42%
Opera 20 és újabbak	1,33%	1,36%	0,96%
Chrome 29 és régebbiek	1,24%	1,13%	0,97%
Opera 19 és régebbiek	1,02%	0,91%	0,72%
Safari 6.0 és újabbak	0,62%	0,63%	0,48%
Safari 5.1 és régebbiek	0,27%	0,26%	0,41%
IE 6-7	0,13%	0,12%	0,20%
mobil böngészok*	12,57%	14,76%	15,44%

* ebbol az Opera Mini-k kb. 0,5- 0,6%

CSS-el különböző tulajdonságokat rendelve a *thead*, *tbody*, *tfoot* címkékhez változatos formázási lehetőségek adódnak. Egy egyszerű példa a fenti szerkezeti egységek szerinti (a CSS részben ismertetendő) formázás eredményeképpeni megjelenítésre:

új táblázat

file:///C:/Users/Orsi/Documents/honlap/új táblázat.html

Böngészők statisztikája 2014. nyarán

hónap	*június*	*július*	*augusztus*
Chrome 30 és újabbak	37,49%	36,61%	36,58%
Firefox 25 és újabbak	32,46%	31,49%	30,94%
IE 10-11	6,72%	7,02%	7,72%
IE 8	2,37%	2,26%	2,31%
Firefox 24 és régebbiek	2,27%	1,99%	1,85%
IE 9	1,51%	1,46%	1,42%
Opera 20 és újabbak	1,33%	1,36%	0,96%
Chrome 29 és régebbiek	1,24%	1,13%	0,97%
Opera 19 és régebbiek	1,02%	0,91%	0,72%
Safari 6.0 és újabbak	0,62%	0,63%	0,48%
Safari 5.1 és régebbiek	0,27%	0,26%	0,41%
IE 6-7	0,13%	0,12%	0,20%
mobil böngészők*	12,57%	14,76%	15,44%

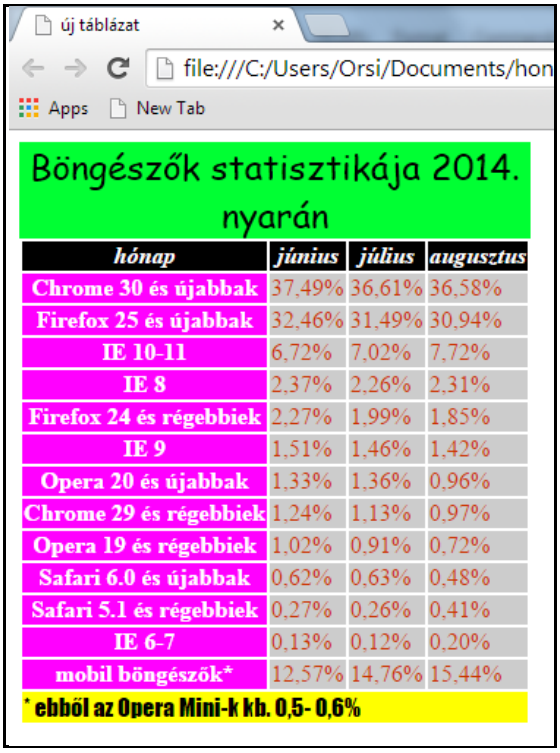
* ebből az Opera Mini-k kb. 0,5- 0,6%

2.6.3. A <*col*> és <*colgroup*> címkékkel a táblázat egy vagy több oszlopa, ill. azokból képzett oszlopcsoportok definiálhatók, és rendelhető hozzájuk pl. formázás. Nem kötelező valamennyi oszlopot bekódolni, csak azokat, melyekre hivatkozni akarunk.

Az oszlopocsoport kódját a *caption* és az első *tr* közé kell elhelyezni (de ha a táblázat végére kódoljuk, akkor is értelmezik a böngészők). Az előző táblázat első oszlopának definiálása tehát:

```
<table>
      <caption>Böngészők statisztikája 2014. nyarán</caption>
      <colgroup><col></colgroup>
      <thead>
            <tr><th>hónap</th><th>január</th><th>február</th>
                <th>március</th>
            </tr>
      </thead>
     .........stb......
</table>
```

Az oszlopokra vonatkozó csoportosító címkék bekódolása nem okoz változást az alapértelmezett megjelenítésben. Ha viszont különböző színű hátteret rendelünk CSS-el a kijelölt oszlopokhoz, a formázott megjelenítés:



Böngészők statisztikája 2014. nyarán

hónap	június	július	augusztus
Chrome 30 és újabbak	37,49%	36,61%	36,58%
Firefox 25 és újabbak	32,46%	31,49%	30,94%
IE 10-11	6,72%	7,02%	7,72%
IE 8	2,37%	2,26%	2,31%
Firefox 24 és régebbiek	2,27%	1,99%	1,85%
IE 9	1,51%	1,46%	1,42%
Opera 20 és újabbak	1,33%	1,36%	0,96%
Chrome 29 és régebbiek	1,24%	1,13%	0,97%
Opera 19 és régebbiek	1,02%	0,91%	0,72%
Safari 6.0 és újabbak	0,62%	0,63%	0,48%
Safari 5.1 és régebbiek	0,27%	0,26%	0,41%
IE 6-7	0,13%	0,12%	0,20%
mobil böngészők*	12,57%	14,76%	15,44%

* ebből az Opera Mini-k kb. 0,5- 0,6%

Megjegyzés: A böngészők csak korlátozottan értelmeznek *col*-hoz rendelt tulajdonságokat, ezért a gyakorlatban kevéssé használatos a *colgroup* -al megvalósított formázás.

Mielőtt befejeznénk a táblázatok kódolásának bemutatását, érdemes elemezni a fenti táblázatok tartalmát is:

- 2013 végén az Internet Explorer 6/7/8 és az Opera Mini-k képezték a HTML5 újdonságainak használatára külön programozási eljárások nélkül alkalmatlan, kb. 4-5 %-os forgalmi részarányt. Ez az érték 2014 nyarára kb. 3-4%-ra csökkent, és év végére valószínű-

leg 2-3%-ra esik, melyen belül az IE 6/7 aránya elhanyagolható lesz. Ezért az IE6 és IE7 említésétől a továbbiakban eltekintünk, a még legalább egy-két évig mérhetően forgalomban maradó IE8-al kapcsolatos problémákat pedig az egyes tárgyalt fejezeteknél említjük meg.

- A CSS3 újdonságokat viszonylag jól (de nem teljeskörűen) értelmezik az IE 10-11, a 6.0-tól a Safari-k és az „új" Opera-k, Firefox-ok és Chrome-ok (részarányuk kb. 78%), ill. mobil verzióik (kb. 7%). Részben értelmezik a mintegy 6% forgalmi részarányt képviselő IE9, a 6.0 előtti Safari-k, és a „régebbi" Chrome-ok, Opera-k és Firefox-ok. Mobil verzióiknak vannak az asztalikhoz képest további kisebb lemaradásaik (és/vagy gyártó-specifikus előtagot ígényelnek a kódolás során), ezek további kb. 6%-ot tesznek ki. A HTML5-re alkalmatlan IE 6/7/8 és Opera Mini-k a CSS3 tulajdonságokat sem alkalmazzák. 2014 végére a verzióváltások eredményeként a CSS3 újdonságokat viszonylag jól értelmező böngészők forgalmi aránya a jelenlegi 85%-ról 90%-ra nőhet (további kb. 7-8%-ot pedig a részlegesen értelmezők teszik ki).

Figyelem! A táblázatok blokkszintű elemek. Alapértelmezettől eltérő megjelenítésük mindig CSS-el történjen. *(lásd a CSS rész Táblázatok formázása fejezetet)*

A TÁBLÁZATOKAT EGY WEBLAP SZERKEZETI KIALAKÍTÁSÁRA FELHASZNÁLNI ELAVULT ÉS NEM JAVASOLT MEGOLDÁS !

A táblázatoknál felhasznált HTML címkék:

Címke:	Funkciója:	
<table>	táblázatot definiál	páros címke
<th>	táblázat fejlécet definiál	páros címke
<tr>	táblázat sort definiál	páros címke
<td>	táblázat cellát definiál	páros címke
<caption>	táblázat címet definiál	páros címke
<thead>	táblázatfejet definiál	páros címke
<tbody>	táblázattörzset defriniál	páros címke
<tfoot>	táblázat láblécet definiál	páros címke
<colgroup>	táblázat oszlopcsoportot definiál	páros címke
<col>	táblázat oszlopo(ka)t definiál	páratlan címke

2.7. Képek

A web-es multimédia tartalmak közül az állóképek kezelése régóta kiforrott, áttekinthetően és egyértelműen szabályozott. A mozgóképekről mindez kevésbé mondható el - azzal a következő fejezet foglalkozik.

Az *img* (img=*image*=kép) címke egy **állókép**et helyez el a weblapon. Kötelező jellemzője az *src* (src=*source*=forrás), ez adja meg a kép elérési útját. Ha a képet a HTML dokumentummal közös gyűjtőmappában tároljuk, forrásként elég a fájlnév és kiterjesztés megadása.

<img src="`fájlnév.kiterjesztés`***">*** páratlan címke

A böngészők a sok képformátum közül a GIF, JPEG és PNG formátumokat tudják értelmezni (a Google által kifejlesztett, de rajta kívül egyelőre csak az Opera által támogatott

WebP, továbbá a Firefox fejlesztésű, és Safari 8 által is támogatott APNG elterjedéséről még korai találgatásba bocsátkozni).

A formátumok főbb tulajdonságainak ismertetése a CSS rész *Képformátumok fejezetében* található.

A kép alapértelmezetten sorban elhelyezkedő (*inline*) elem (folyamatosan, a sorban előtte lévő elemet követi, ill. a következő elem folytatólagosan halad a sorban tovább), és nincsen kerete. Blokkszintű elemmé később tárgyalt módon alakítható, formázása és egyéb pozícionálása pedig CSS-el történik (lásd a CSS részben a *Szegélyek* és a *Blokkszintű elemek elhelyezése I.-II.* fejezeteket).

Az *img* címke további, ajánlott jellemzői:

a) A *width* (szélesség) és *height* (magasság) nem kötelező jellemzők, de segítenek a böngészőknek már a kép letöltődése előtt elrendezni a megjelenítést (azonnal el tudják helyezni a szöveget anélkül, hogy meg kellene várniuk a képek letöltődését), így nem ugrik a részben megjelent oldal a kép(ek) később ismertté vált mérete(i) miatt.

Ha a kép fizikai méretétől eltérő szélességet és magasságot adunk meg, nagyítás, kicsinyítés, vagy (tudatos vagy figyelmetlenségből elkövetett) torzítás következik be. A nagyítás vagy kicsinyítés nem érinti a kép felbontását – kicsinyítéskor nem fog gyorsabban letöltődni, nagyításkor pixelesedés léphet fel.

A szélesség és magasság kódolása:

```
<img src="fájlnév.kiterjesztés" width="…" height="…">
```

b) Az *alt* jellemző (alt=*alternative text*=helyettesítő szöveg) képet helyettesítő szöveges információ. Csak akkor látszódik, ha a kép nem jelenik meg (nem töltődik le, nem érhető el, stb.), célszerűen a kép tartalmát, célját ismerteti a kép helyettesítéseként.

```
<img src="fájlnév.kiterjesztés" alt="…">
```

Ha a képpel egy gombot alakítunk ki, helyettesítő szövegként a gomb feliratát érdemes megadni. Ha az *alt* jellemzőt *alt=""*-ként beírjuk, (nem adunk meg értéket - még szóközt sem - az idézőjelek között), azt jelezzük, hogy a kép nem a tartalom szerves része, csak ahhoz lazán kapcsolódik.

Figyelem ! Ha a kép teljes mértékben csak díszítőelem, akkor nem képként, hanem háttérképként kell megadni. A háttérképek CSS-el alakíthatók ki, lásd a CSS rész *Háttérképek definiálása* fejezetet.

c) A *title* jellemző (*title*= cím, elnevezés, felirat) olyan szöveges információt definiál, melyet csak akkor mutat meg a böngésző, ha a megjelenített kép fölé állunk az egérrel (*tooltip* = rövid súgó szöveg, elemleírás, buborékszöveg) - így ez nyomtatásban nem látszódik.

Valamennyi felsorolt jellemzőt felhasználva egy kép HTML kódja:

```
<img src="fájlnév.kiterjesztés" width="…" height="…" alt="……"
title="……">
```

Példaként egy szövegben elhelyezünk egy képet: „A rózsa – ide beillesztjük egy rózsa képét ”sok vörös rózsa” helyettesítő szöveggel és az egérrel a képre állva megjelenő „rózsák” felirattal – a rózsafélék családjának egyik nemzetsége, nemesített fajait virága, olaja miatt termesztik.”

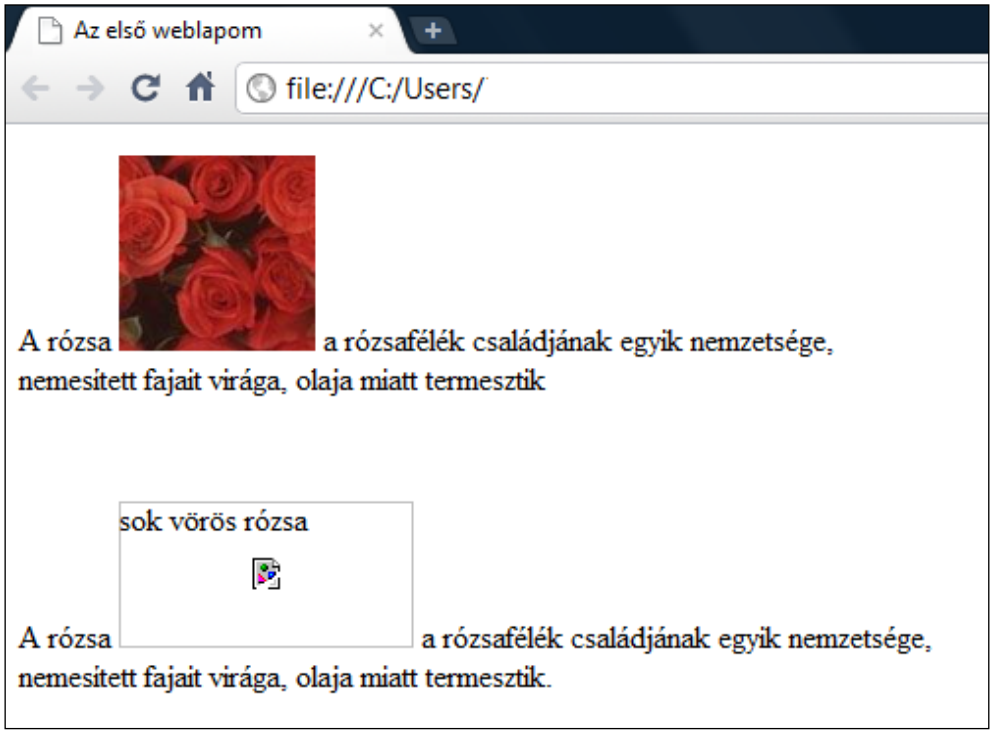
A kódolás jó képfájlnévvel és torzítatlan képaránnyal a következő lesz:

```
<p>A rózsa<img src="piros.jpg" width="100px" height="100px"
alt="sok vörös rózsa" title="rózsák"> a rózsafélék családjának
egyik nemzetsége,<br> nemesített fajait virága, olaja miatt
termesztik.</p>
```

Torzítva a kép arányait és tudatosan rossz fájlnevet adva, hogy a kép ne tudjon letöltődni:

```
<p>A rózsa <img src="vörös.jpg" width="150px" height="75px"
alt="sok vörös rózsa" title="rózsák"> a rózsafélék családjának
egyik nemzetsége,<br> nemesített fajait virága, olaja miatt
termesztik.</p>
```

A két eset megjelenítése az alábbi lesz:



Figyelem! Egy kép lehet lista eleme, táblázat eleme, és a későbbi fejezetben tárgyalt hivatkozásokban a szöveg mellett vagy helyett alkalmazott elem is.

<u>A képeknél felhasznált HTML címkék:</u>

Címke:	Funkciója:	
<image>	(álló)képet definiál	páratlan címke

2.8. Mozgóképek és hang

A mozgóképek (videó, animáció) és hang weboldalon történő megjelenítésére ill. megszólaltatására alapvetően négy alternatív lehetőség van. Ebből kettőt ebben a fejezetben, egyet a *Beágyazott böngészőtartalom*, egyet pedig a *Hivatkozások* fejezetben tárgyalunk.

2.8.1. A „HTML5 előtti kor" technikájában az *<object>* (idegen objektumok beágyazására szolgáló) páros címkével lehet egy weblapra multimedia - Flash, Silverlight, Windows Media, RealPlayer, Quick Time, stb. – fájlokat feltenni (az állóképet mint multimédia elemet már korábban külön tárgyaltuk). A böngészők egységes értelmezése céljából még az *<embed>* (egyébként a HTML5-ig nem szabványosított) címkével is szokás kiegészíteni a kódolást.

A fájlok lejátszásához a felhasználónak rendelkeznie kell a megfelelő (ingyenesen letölthető) bővítménnyel. A böngészők ugyanis az ilyen multimédiás tartalmat egy (általában másik cég által készített) bővítménybe (*plug-in*) rakják, ami a böngésző számára egy „fekete doboz", a multimédia-fájlhoz nincsen hozzáférése. A bővítményt vagy a böngésző adja, vagy a felhasználónak magának kell letöltenie, és gondoskodnia a bővítmény és böngésző kompatibilitásáról.

Az *<object>……</object>* elem jellemzői és azok értékei:

- *classid* (class identifyer = osztályazonosító) a lejátszáshoz szükséges bővítményt definiálja, értéke egy nagyon hosszú karaktersor, például a QuickTime esetében clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B, a Windows Media Player-nél clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6, stb.
- *codebase* (kódalap) a lejátszó változatának letöltési helye, pl. http://www. microsoft. com/ windows/windowsmedia, stb.
- *width* és *height* a szélesség és magasság méretek (az állóképekkel analóg módon)

A lejátszás egyéb paramétereit a *param* páratlan címkének a jellemzőivel/értékeivel kell megadni, pl.:

```
name="type" value="video/quicktime"          egy QuickTime video-t ad meg
name="URL" value="fájlnév.kiterjesztés"      a médiafájl elérhetősége
name="autoplay" value=""                     elinduljon-e automatikusan a lejátszás
name="controller" value=""                   legyenek-e kezelőszervek a lejátszáshoz
stb.
```

A beágyazás kódolása tehát:

```
<object classid="……" codebase="……" width="……" height="……">
    <param name="……" value="……">
    <param name="……" value="……">
    <param name="……" value="……">
    <param name="……" value="……">
</object>
```

A YouTube-ról vett tartalom esetén a beágyazó kódolás a *video*-ra jobb egérgombbal kattintva és a *copy embed code* menüpontot választva bemásolható. Találomra kiválasztva egy videót a felkínált beágyazó kódsor:

```
<object style="height:390px; width:640px">
      <param name="movie" value="http://www.youtube.com/v/3IwD
             HvNxjJo?version=3&feature=player_detailpage">
      <param name="allow FullScreen" value="true">
      <param name="allowScriptAccess" value="always">
```

```
<embed src="http://www.youtube.com/v/3IwDHvNxjJo?version=3&
  feature=player_detailpage" type="application/x-shockwave-
  flash" allowfullscreen="true" allow ScriptAccess="always"
  width ="640" height="360">
</object>
```

2.8.2. A HTML5 készülő szabványa módot nyújt a hang és mozgókép közvetlenül a böngészőkben (bővítmény igénybevétele nélküli) lejátszására, manipulálására, CSS-el való formázására. Bár az <*audio*> és <*video*> címkéket már a Firefox 3.5, Opera 10.5, Chrome 3.0, Internet Explorer 9 és Safari 3.0 óta tartalmazzák a böngészők, a funkciók még a szabványosítás folyamatában vannak.

A mozgókép-fájl egy videosávot, egy vagy több audiosávot és egy vagy több feliratozássávot tartalmazhat. Mindezeket egy *container* formátum rendezi egy video-fájlba.

Rengeteg *container* formátum létezik, a legelterjedtebbek:

ASF: kiterjesztése *.asf* vagy *.wmv* (Microsoft)
AVI: kiterjesztése *.avi* (Microsoft)
Flash Video: kiterjesztése *.flv* (Adobe)
Matroshka: kiterjesztése *.mkv* (CoreCodec)
MPEG-4: kiterjesztése általában *.mp4* (MPEG LA – több mint két tucat társaság, Microsoft-ot és Apple-t is beleértve)
Ogg: kiterjesztése *.ogv* (Xiph.Org Foundation)
QuickTime: kiterjesztése *.mov* (Apple)

A mozgókép tartalmat a *video codec* (codec=coder-decoder) kódolja-dekódolja és tömöríti (a világhálón veszteséges tömörítési algoritmusokat használnak). Rengeteg veszteséges video codec létezik, a legelterjedtebbek:

Dirac: beágyazható az *.mp4*, *.ogv*, *.mkv* és *.avi container*-ekbe
H.264: beágyazható az *.mp4* és *.mkv container*-ekbe
MPEG-4 ASP: beágyazható az *.avi*, *mp4* és *.mkv container*-ekbe
Theora: bármelyik *container*-be beágyazható, de leggyakrabban az *.ogv*-ban használják
VC-1: elvileg *container*-független, gyakorlatilag az *.asf container*-ben használják
VP8: a Google által favorizált codec (ill. utódja, a VP9 - lásd lejjebb)

A hang tartalmat az *audio codec* kódolja-dekódolja és tömöríti (a világhálón itt is veszteséges tömörítési algoritmusokat használnak). Rengeteg veszteséges audio codec létezik, a legelterjedtebbek:

AAC: max. 48 hangcsatorna, *.mp4 container*-be beágyazható
MP3: max. 2 hangcsatorna, bármelyik *container*-be beágyazható
Vorbis. tetszőleges számú hangcsatorna, általában *.ogv*, ritkábban *.mp4* vagy *.mk* container-ben alkalmazzák

Az IETF (Internet Engineering Task Force) 2012 őszén a Skype SILK és a Xiph.Org CELT-jén alapuló *OPUS*-t fogadta el webes hangtömörítési szabványként, de ezt egyelőre csak a Firefox, Chrome és Opera támogatja. A nyílt forráskódú, szabad felhasználású, jó hangminőséget biztosító *OPUS*-nak a jogdíjas MP3-at kéne leváltania.

A feliratozás a *track* címkével, a W3C szabványosítását elkerülő WebVTT (Web Video Text Tracks) nyelven történik (a *track* címkét - legutolsóként - a Firefox 31 vezette be).

A HTML5 <*video*> címkénél a W3C nem ad sem a *container*-re, sem az *audio codec*-re vagy *video codec*-re előírást. Tehát egy <*video*> elemhez több video-fájl kell hogy tartozzon, és a böngésző az elsőt fogja választani, amit le tud játszani. Azt pedig a felhasználóknak kell tudniuk, hogy melyik böngésző melyik *container(ek)*-t és *codec(ek)*-et támogatja.

Jelenleg az alábbi három versengő rendszer (container + video + audio) van:

.mp4	- H.264 + AAC	(Microsoft, Apple – jelenleg ez a legelterjedtebb)
.ogv	- Theora + Vorbis	(Firefox, Google, Opera – eredetileg ezt szánták W3C szabványnak)
.webm	- VP8 + Vorbis	(Firefox, Google, Opera – ezt, ill. utódját, a VP9-et tervezi a Google elterjeszteni)

A Mozilla, Google és Opera „WebM Project" néven dolgozik a *webm* közös formátum kialakításán, melynek fő előnye a szabad felhasználás lesz. Az Apple és Microsoft az *.mp4 – H.264 – AAC* –re koncentrál (a H.264-et tekintik a minőség/fájlméret szempontjából a legjobb video-formátumnak, de szabadalmi joggal védett). Tartalékként azért a konkurens rendszerek felé is nyitva maradnak ajtók, pl. ha a felhasználó külön letölt magának egy *VP8 codec*-et, akkor a WebM-et is használhatja az Internet Explorer 9-től, a Microsoft pedig készített egy MPEG-4 bővítményt a Firefox-hoz. A Chrome úgy támogatja az MPEG-4-et, hogy bejelentette, idővel majd felhagy vele - de az időpontot nem közölte.

A képet tovább bonyolítja, hogy egyes gyártók asztali és mobil böngészői eltérő rendszert támogatnak, pl.:

- Google: Ogg/Theora és Opus van az asztali Chrome-ban, de nincsenek az Android és Chrome for Android mobil böngészőkben
- Opera: Ogg/Theora és Opus van az asztali Opera-ban, de nincsenek az Opera Mobile böngészőben
- Opera: MPEG-4/H.264 nincsen az asztali Opera-ban, de van az Opera Mobile-ban
- A Firefox az MPEG-4/H.264-et részlegesen (Windows 7 és Windows 8.x –en) támogatja.

Megjegyzés: Az Internet Explorer 8 egyáltalán nem ismeri fel a *video* címkét.

Figyelem! Jelenleg nincsen egyetlen olyan *container* és *codec* kombináció sem, amely valamennyi - egyébként a HTML5 *video* funkciót értelmező - böngészőben működne. Ahhoz, hogy a mozgókép bármely ilyen böngészőben megnézhető legyen, többszörösen kell kódolni.

A mozgóképet elvileg az állókép <*img*> címkéjéhez hasonló struktúrájú kódolásban – de páros címkével - tervezték, tehát (ha a gyűjtőmappában van a video-fájl is):

```
<video src="fájlnév.kiterjesztés"></video>
```

A szélességet és magasságot itt is erősen javasolt jellemző/értékként megadni, az állóképhez képest jelentős különbség viszont, hogy itt a lejátszás körülményeit is definiálni kell.

A *preload* (előre letöltés) jellemző lehetséges értékei:

- *none*: automatikusan semmilyen letöltés megkezdését nem engedélyezi
- *metadata*: csak a video tulajdonságai töltődnek le automatikusan
- *auto*: a teljes tartalom automatikusan letöltődik

Az *autoplay* (lejátszás automatikus elindulása) jellemzőnek nem kell értéket adni, amennyiben beírjuk a HTML kódba, automatikusan lejátszást kezdeményez.

A *loop* jellemzőnek ugyancsak nem kell értéket adni, a jellemző megadása esetén a lejátszás végét követően megismétlődik (újrakezdődik) a lejátszás.

Fontos! Az *autoplay* és *loop* jellemző és a *preload auto* érték használata kódolás szempontjából ugyan helyes, használata azonban nem javasolt – a felhasználó döntése kell hogy legyen, hogy meg akarja-e nézni a *video*-t, s ha igen, akkor hányszor, ill. kifizeti-e a letöltés adatforgalmi díját – nem illik kéretlenül rázúdítani egyiket sem.

A *poster* jellemzővel a címlapkép adható meg, értéke a kép elérési útja (azonos gyűjtőmappa esetén a fájlnév és kiterjesztés), azaz

```
poster="fájlnév.kiterjesztés".
```

A *controls* jellemzőnek nem kell értéket adni, definiálása esetén a lejátszást szabályozó vezérlőgombokat a böngésző mellékeli a *video*-hoz (ha nem adjuk meg, akkor vagy nem tudjuk kezelni a *video*-t, vagy szkript-nyelven magunknak kell legyártani egy vezérlést). Az eszközpanel kialakítása böngészőfüggő.

A kiinduló *video*-kódolást a fenti jellemzőkkel kiegészítve már némileg hosszabb HTML-kód adódik:

```
<video src="fájlnév.kiterjesztés"
     poster="fájlnév.kiterjesztés" controls preload="metadata"
     width="……" height="……">
</video>
```

Figyelem! A *source* és *poster* kiterjesztése természetesen eltérő, hiszen az állókép valószínűleg PNG vagy JPG, a *video* pedig valószínűleg WebM, Ogg vagy MPEG-4.

Tekintettel a *container*-ek és *codec*-ek változatosságára, a *source* jellemző helyett *source* címkét szokás használni a *video* elem tartalmának meghatározására. Ekkor az eredeti struktúra (<***video*** ***src***=*"fájlnév.kiterjesztés"*><***/video***>) többszörös fájlformátumú kódolása:

```
<video>
    <source src="fájlnév.kiterjesztés1">
    <source src="fájlnév.kiterjesztés2">
    <source src="fájlnév.kiterjesztés3">
</video>
```

Valamennyi HTML5 formátummal való kompatibilitást megcélozva a mozgókép egy verziójának WebM (VP8 + Vorbis)-ban, egy második verziónak MP4 (H.264 baseline video + AAC “low complexity” audio)-ban, egy harmadik verziónak Ogg (Theora + Vorbis)-ban kell a webhelyre kerülnie. Az egyes formátumoknak a *type* jellemzővel történő (tulajdonképpen ismételt) megadásának a célja, hogy a böngésző egyből ki tudja választani a számára értelmezhető verziót, ne pedig sorban próbálgatva jusson el hozzá.

```
<video width="……" height="……" controls preload="metadata">
    <source src="….mp4" type='video/mp4; codecs="avc1.42E01E,
            mp4a.40.2"'>
    <source src="…….webm" type='video/webm; codecs="vp8,
            vorbis"'>
    <source src="…….ogv" type='video/ogg; codecs="theora,
            vorbis"'>
</video>
```

Figyelem! Mivel a *codecs* értékét dupla idézőjelben kell megadni, ezért a *type* értéke szimpla idézőjelbe került. Ha a *type*-ban a codec-et nem definiáljuk, maradhat a szokásos dupla idézőjeles értékmegjelölés.

A <*video*> elem megadja a méreteket és a lejátszással kapcsolatos adatokat, de nem kapcsolódik közvetlenül a video-fájlhoz. A belsejében három <*source*> elem van, ezek a *src* jellemzőjükkel kapcsolódnak az egyes video-fájlokhoz, a *type* jellemzővel megadva a *container* formátumot és a *codec*-eket.

2.8.2. A <*video*> címkéjéhez hasonlóan használható az <*audio*> címke, például a legegyszerűbb esetben:

```
<audio src="fájlnév.kiterjesztés"></audio>
```

Természetesen nincsen *width*, *height* és *poster* jellemző, egyébként mindenben a *video*-nál leírtak szerint kell kódolni (*autoplay*, *loop*, *preload* és *controls* jellemzők definiálása, többszörös hangfájl-formátum kódolása, stb.):

```
<audio controls preload="……">
    <source src="fájlnév.kiterjesztés1">
    <source src="fájlnév.kiterjesztés2">
</audio>
```

Az MP3 jelenleg de facto szabvány, mellette a *.ogg* vagy *.wav* kiterjesztés fordulhat még leginkább elő. A video-nál tett, *autoplay*-el és *loop*-al kapcsolatos aggályok/észrevételek itt is érvényesek.

A HTML5-alapú audio és video terjedése ellenére tartalékként nem-HTML-alapú (leggyakrabban Flash) lejátszást is szokás biztosítani. Ebben az esetben a kódolást az <*object*> -nél látottakkal kell bővíteni.

Megjegyzés: A Flash szerepe a közeljövőben várhatóan változáson megy keresztül:

- Az Adobe cég 2011. végével lezárta a mobil platformokra történő Flash Player fejlesztéseket. Közleménye szerint: „Az elmúlt két év során a mobil böngészőkhöz is elérhetővé tettük a Flash Player szoftvert, azonban mára minden jelentős mobileszköz támogatja a HTML5 technológiát, gyakran kizárólagosan. Minden mobil (táblagép és okostelefon) platformon a HTML5 a legjobb megoldás a tartalmak létrehozásához és böngészőkben való terjesztéséhez. Nem fejlesztjük tovább a böngészőkben használható Flash Player szoftvert az új mobileszköz konfigurációk támogatására.”

- Bár az Android 4.0-ra még rendelkezésre áll a Flash Playert, a Google 2012 augusztusától megszüntette – valamennyi, tehát a régebbi Android verziókhoz is – a Google Play-en keresztül a Flash Player letölthetőségét

- az Apple kezdettől fogva elzárkozott az iOS/SafariMobile-ban a Flash támogatásától

- A YouTube-on próbaüzemben WebM-ben is lehet már egy ideje videókat kezelni, 2014 május végétől pedig alapértelmezésben a HTML5-ös lejátszót tölti be a YouTube a Chrome böngészőbe. Tekintettel a YouTube piaci súlyára, és hogy a weblapok multimédia tartalma kezelésének egyik bevett módja is a YouTube-on „keresztül futtatás”, a Flash-ről WebM-re való áttérés kérdése a Flash használatának elterjedtségét komolyan befolyásolhatja.

- a HTML5 Encrypted Media Extensions (EME) fejlesztése lehetővé teszi a szerzői joggal védett film, zene és tv plug-in nélküli alkalmazását.

Megjegyzés: Microsoft is abbahagyta a Silverlight fejlesztését, bár 2021-ig biztosítja a támogatását.

A mozgóképeknél és hangnál felhasznált HTML címkék:

Címke:	Funkciója:	
<video>	mozgóképet definiál	páros címke
<audio>	hangot definiál	páros címke
<source>	multimédia elemet definiál	páros címke
<object>	beágyazott multimédia elemet definiál	páros címke
<embed>	beágyazott multimédia elemet definiál	páros címke
<param>	multimédiás objektum beállításait definiálja	páratlan címke
<track>	video feliratozást tartalmaz	páros címke

2.9. Beágyazott böngészőtartalom

Egy weblapba az *<iframe>……..</iframe>* páros címkével (iframe = *inline frame* = szövegközi/beágyazott keret) egy másik weblapról böngészési tartalom illeszthető be. Az állóképhez hasonlóan *inline* elemként viselkedik, és amennyiben nem fér el a böngészett tartalom a számára kijelölt helyen, automatikusan megjelenő görgetősávok segítségével nézhetők meg a nem látszó részek. A címke kötelező jellemzője a böngészett tartalom forrása, melynek értéke a webhelynek a teljes URL-je – és az álló- vagy mozgóképhez hasonlóan erősen javasolt jellemzője a *width* és a *height*.

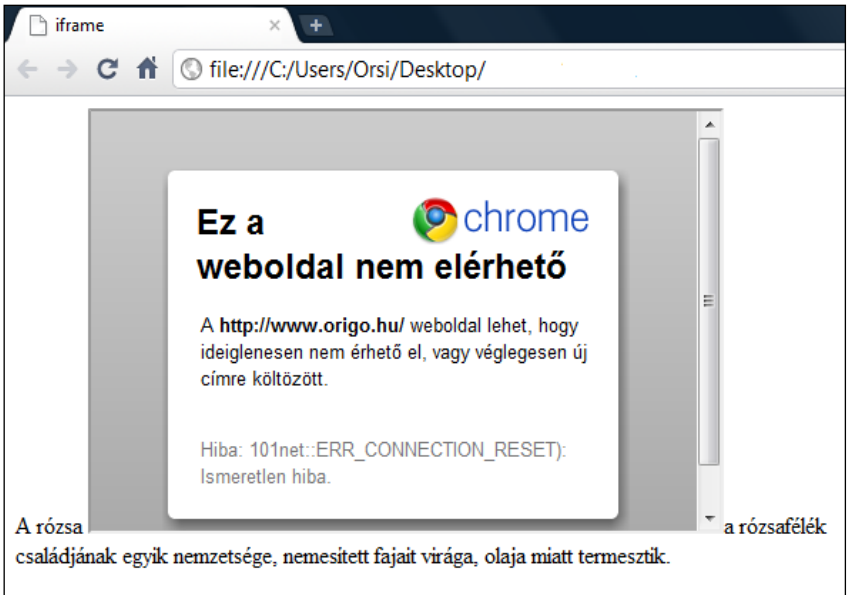
Figyelem! Míg böngészéskor általában elég a *www.* kezdettel (vagy sokszor anélkül is) beírni egy oldal URL-jét, a HTML-kódolás során a teljes URL a *http://*-t is kötelezően magában foglalja.

A rózsás képünk helyére beillesztve pl. az *origo.hu* nyitólapját a kódolás az alábbi:

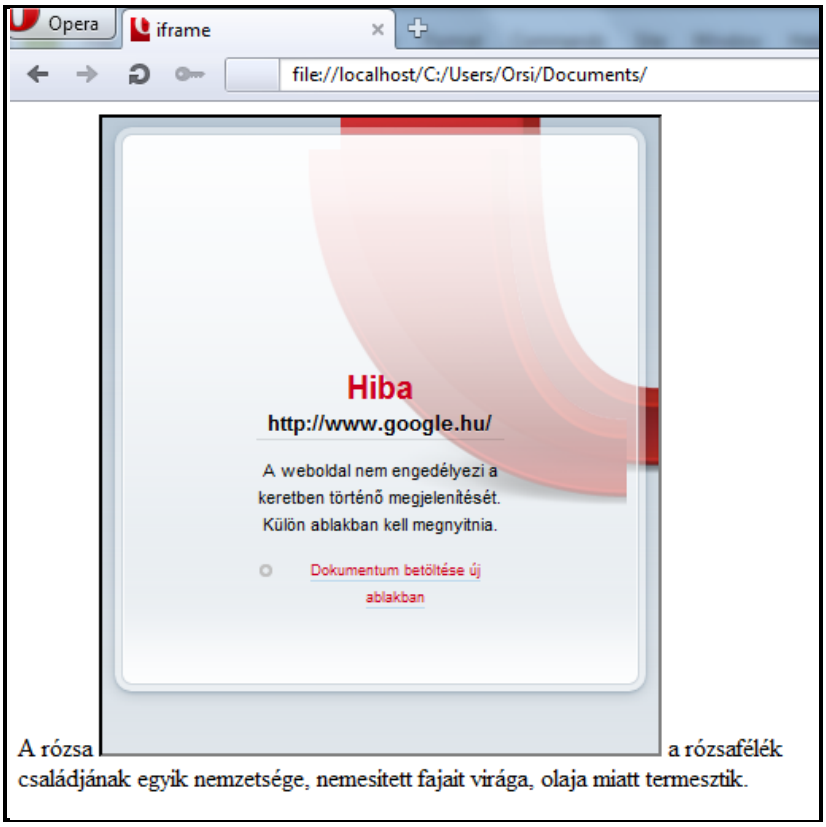
```
<iframe src="http://www.origo.hu" width="35%" height="40%">
</iframe>
```

Alapértelmezetten a megjelenítés (mely CSS-el tovább formázható):

Az *iframe* címke csak on-line üzemben működik rendeltetésszerűen – ha nem érhető el a megadott URL, akkor a keresett böngészőtartalom helyett a megjelenítés:



Bizonyos webhelyek nem engedélyezik a keretben való megjelenítést, ekkor csak külön ablakban – a felkínált *link*-re kattintva - lehet a felkeresett webhelyet használni:



Megjegyzés: A beágyazott böngészőtartalmat a keresők nem veszik figyelembe (más tollával nem lehet ékeskedni).

Nem kötelező jellemzője az *iframe* címkének a *sandbox* (a *sandbox* itt virtuális karantént jelent), mellyel az *iframe*-ben való böngészés során a biztonság növelése céljából kizárhatóak ill. megengedhetőek bizonyos kódok.

A *sandbox* lehetséges értékei: *allow-same-origin, allow-top-navigation, allow-forms, allow-scripts* – ezeket szóközzel, vessző nélkül felsorolva lehet megadni. Pl.:

```
<iframe src="http://www.origo.hu" width="35%" height="40%"
        sandbox="allow-scripts allow-forms">
</iframe>
```

Egy **video**-nak a weblapba történő **beágyazásá**hoz az egyik módszer szintén az *iframe* alkalmazása. A YouTube-ra feltöltött vagy már ott lévő video az

```
<iframe src="http://www.youtube.com/embed/....  " width="……"
        height="……"
</iframe>
```

kódolással, a YouTube által adott 11-jegyű kód átmásolásával a weblapon lejátszható. Nem kell bajlódni a video-formátumokkal és böngészőverziókkal – sőt a *share – embed* kattintásokkal a teljes *iframe* elem bemásolható.

Egy találomra kiválasztott video másolható kódja:

```
<iframe width="560" height="315" src="http://www.youtube.com/
        embed/BjByTjOkyi0" frameborder="0" allowfullscreen>
</iframe>
```

A beágyazott böngészőtartalomnál felhasznált HTML címkék:

Címke:	Funkciója:	
<iframe>	szövegközi keretet (beágyazott böngészőtartalmat) definiál	páros címke

2.10. Karakter entitások

A HTML dokumentum tartalmának kialakítása során a foglalt karakterek ill. a billentyűzetről nem, vagy nehezen elérhető különleges karakterek esetében un. karakter entitásokat (*character entity*) kell alkalmazni, ami az adott karakter/szimbólum/jel kódolt megadását jelenti.

Foglalt karakterek: a kódolásban használt néhány karakternek a folyó szövegben való használata a böngészőt megtévesztheti és HTML címkének tekintheti. Ezek:

< kisebb mint (less-than)
> nagyobb mint (greater-than)
” idézőjel (quotation mark)
& és (ampersand)

Billentyűzetről nem (vagy nehezen) bevihető **különleges karakterek:** görög betűk, pénznemek (yuan, yen, font, stb.) matematikai szimbólumok (végtelen, gyök, összesen, közel egyenlő, azonos), stb.

A foglalt és különleges karakterek kódja két módon is megadható: entitás számmal (numeric characters), vagy entitás névvel (named characters). Mindkét megoldásnál a kód &-vel kezdődik és ;-vel végződik, az entitás számnál # és szám, az entitás névnél szöveg van a kód belsejében. A # (kettőskereszt) hivatalos elnevezése „oktotorp”, de az egyszerűség kedvéért a továbbiakban a kettőskereszt kifejezést használjuk.

Figyelem! Az entitás számmal történő megadása történhet tízes (decimális) vagy tizenhatos (hexadecimális) számrendszerben is – a felsorolásunkban csak a decimális karakterkódot tüntettük fel. Az entitás névvel történő megadása kisbetű-nagybetű érzékeny.

A foglalt karakterek és néhány gyakrabban használt különleges karakter entitásai:

karakter	entitás szám	entitás név	angol megnevezés
<	<	<	(less-than)
>	>	>	(greater-than)
”	"	"	(quotation mark)
&	'	&	(ampersand)
©	©	©	(copyright)
®	®	®	(registered trademark)
™	™	™	(trademark)
°	°	°	(degree)
€	€	€	(euro)
¢	¢	¢	(cent)
§	§	§	(section)
±	±	±	(plus-or-minus)
‰	‰	‰	(per mille)
x	×	×	(multiplication)
			(non-breaking space)

A karakter entitások teljes skálája a W3C webhelyén nézhető meg.

Az utolsó karakter entitáshoz tartozik magyarázat: a **nem törhető szóköz**nek (nbsp = *non-breaking space*) nincsen megjelenési formája - az a funkciója, hogy több szóból álló kifejezés vagy információ esetén a szavak közé helyezve megakadályozza azt, hogy a sor végén a böngésző a kifejezés/információ közben törje meg a sort. Tipikus példa, hogy pl. a Coca Cola, Túró Rudi, Hong Kong, stb. két szava ne váljon el egymástól.

Ha például a brazil labdarúgó Pelé eredeti nevét (Edson Arantes do Nascimento) úgy akarjuk megjeleníteni, hogy ne kerülhessen több sorba megtörve, akkor a kódolás:

```
Edson Arantes do Nascimento
```

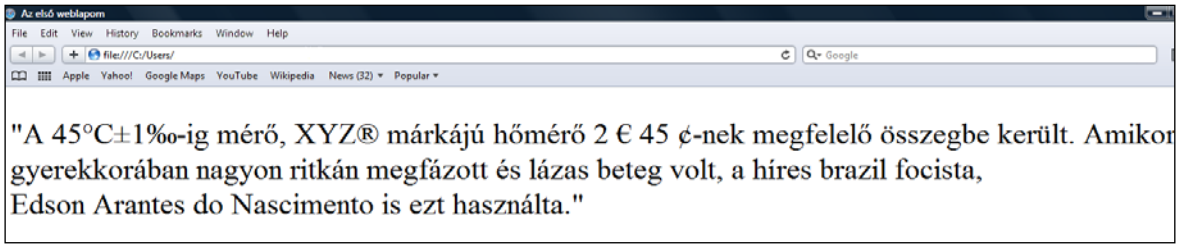
A fentiek alapján egy különleges karakterekkel megtűzdelt mondat kódolása a híres ember neve közbeni sortörést meg nem engedve így néz ki:

„A 45°C ±1‰-ig mérő, XYZ® márkájú hőmérő 2€ 45¢-nek megfelelő összegbe került. Amikor gyerekkorában nagyon ritkán megfázott és lázas beteg volt, a híres brazil focista, Edson Arantes do Nascimento is ezt használta.”

```
<p>&quot;A 45&deg;C&plusmn;1&permil;-ig mérő, XYZ&reg; márkájú
   hőmérő 2 &euro; 45 &cent;-nek megfelelő összegbe került.
   Amikor gyerekkorában nagyon ritkán megfázott és lázas beteg
   volt, a híres brazil focista, Edson Arantes do
     Nascimento is ezt használta.
&quot;</p>
```

(Nyilván nem a mondanivalója, hanem a sok különleges karakter indokolja a fenti példát.)

A böngésző ablaka szélességének folyamatos változtatásával figyelhető meg, hogy a sortörés hogyan változik a hosszú névnél a mondatban (ott nem törik meg):



2.11. Tartalmi/formázási HTML címkék

A HTML címkék egy csoportja, melyeknek korábban a formázásban volt szerepe, a CSS alkalmazása óta olyan módon használatos, hogy bár a megjelenítésre is hatnak, elsősorban tartalmi okok miatt használják őket, és a végső formázást továbbra is CSS-el lehet előállítani. Tartalmi megfontoláson alapuló használatuk azért is célszerű, mert a keresők a HTML címkékkel való kialakítás alapján tudják jobban a találatok közé beilleszteni az ezeket az elemeket tartalmazó weblapokat.

a)-b) **Bizonyos karakterek megjelenítéséhez** is HTML címkékre van szükség. A *_{* ……*}* páros címke (*subscript*=alsó index) pl. elengedhetetlen a vegyjelek írásához (lásd H_2O), a *^{*……*}* páros címke (*superscript*=felső index) pl. idegen kifejezések (lásd az angol 1st, francia 1ere), védjegyek esetén, vagy mértékegységek használatakor (pl. m^2, cm^3) szükséges. (Az előzőekben kódolt XYZ® is helyesen XYZ® kellett volna hogy legyen.)

Például: „Az LMN™ kémcsőbe 5 cm^3 H_2SO_4 került.” mondat kódolva:

```
<p>&quot;Az LMN&trade; kémcsőbe 5 cm<sup>3</sup> H<sub>2</sub>
        SO<sub>4</sub> került.
&quot;</p>
```

c)-d) **Idézetek:** külön páros címke van a rövid idézetekre, és külön a hosszabb, tömbszerű idézetekre.

A rövid, általában mondaton vagy bekezdésen belüli (*inline*) idézetet kódolással a *<q>*……*</q>* párral (*quote*=idézet), a hosszabb, blokkszerű idézeteket a *<blockquote>*…… …*</blockquote>* (*blockquote*=idézetblokk) párral lehet létrehozni.

- Példa a *<quote>* címke használatára:

A világ egyik legismertebb idézete a Hamlet-ból a „Lenni, vagy nem lenni, ez itt a kérdés”.

Kódolva:

```
<p>A világ egyik legismertebb idézete a Hamlet-ból a
   <q>Lenni, vagy nem lenni, ez itt a kérdés.</q>
</p>
```

- Példa a *<blockquote>* címke használatára:

Az ENSZ Emberi Jogok Egyetemes Nyilatkozata első két cikkelye kimondja:

„Minden emberi lény szabadon születik és egyenlő méltósága és joga van. Az emberek, ésszel és lelkiismerettel bírván, egymással szemben testvéri szellemben kell hogy viseltessenek.

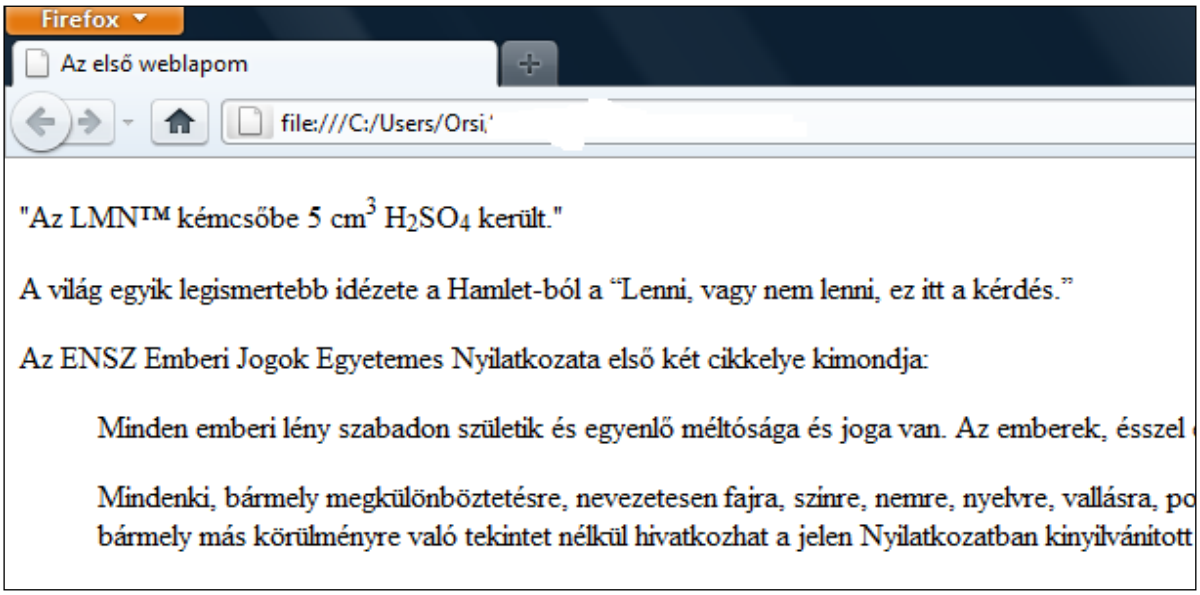
Mindenki, bármely megkülönböztetésre, nevezetesen fajra, színre, nemre, nyelvre, vallásra, politikai vagy bármely más véleményre, nemzeti vagy társadalmi eredetre, vagyonra, születésre, vagy bármely más körülményre való tekintet nélkül hivatkozhat a jelen Nyilatkozatban kinyilvánított összes jogokra és szabadságokra."

Kódolva:

```
<p>
 Az ENSZ Emberi Jogok Egyetemes Nyilatkozata első két cikkelye
 kimondja:
   <blockquote>
      <p> Minden emberi lény ……</p>
      <p> Mindenki, bármely megkülönböztetésre ……</p>
   </blockquote>
</p>
```

A böngészők idézőjel nélkül, mindkét oldali behúzással, előtte és utána térközzel, blokkszintű elemként jelenítik meg az idézetblokkokat, melyben lehetnek beágyazott elemek is (pl. jelen esetben bekezdések). Mindkét idézettípus CSS-el tovább formázható *(lásd CSS rész Szöveg fejezete).*

Az a)-b)-c)-d) címkék megjelenítése:



e) A **rövidítés** páros címkéje az *<abbr>……</abbr>* (*abbreviation*=rövidítés). A rövidítés vagy mozaikszó első előfordulásakor szokás így a teljes alakot feltüntetni, mely csak a képernyőn, a kurzornak a rövidítésre/mozaikszóra állításával egy kis ablakban (*tooltip* = elemleírás, rövid súgó szöveg, buborékszöveg) jelenik meg, nyomtatásban nem látszik. A *title* jellemző (mely csak formailag egyezik meg a *title* páros címkével!) segítségével, annak értékeként adható meg a teljes alak.

Pl. az előző blokkidézet példájára visszatérve, az ENSZ rövidítés teljes alakját megmutatva:

```
<p>Az<abbr title="Egyesült Nemzetek Szervezete">
   ENSZ</abbr> Emberi Jogok Egyetemes Nyilatkozata első két
   cikkelye kimondja:
</p>
```

(Az * * nem törhető szóközt azért kellett alkalmazni, hogy ne csak a *tooltip* első szavát jelenítsék meg a böngészők.)

A Firefox pontozott aláhúzással jelzi, hogy a teljes alak is megtekinthető, a többiek nem alkalmaznak alapértelmezett figyelemfelkeltő megjelenítést. Ha azt kívánjuk, hogy valamennyi böngésző egyformán jelenítse meg, ill. egyformán szembeötlő legyen a további információ elérhetősége, CSS-el hozandó a kívánt formára a rövidítés.
(Lásd CSS rész Betűtípusok és Szöveg fejezeteit)

f) A **meghatározás** *<dfn>……..</dfn>* páros címke (*definition* = meghatározás) a rövidítéssel megegyező módon működik.

Pl. „ Az agyvíz, latinul liquor cerebrospinalis, fontos szerepet játszik az agyszövet anyagcseréjében és a kórokozók elleni védelemben.” mondatban ha megadjuk az agyvíz meghatározását („az agyat és a gerincvelőt borító hártyák közötti résben áramló folyadék”), a HTML kódolás az alábbi lesz:

```
<p>Az<dfn title="az agyat és a gerincvelőt
    borító hártyák közötti résben 
   áramló folyadék"> agyvíz</dfn>, latinul liquor
   cerebrospinalis, fontos szerepet játszik az agyszövet
   anyagcseréjében és a kórokozók elleni védelemben.
</p>
```

(Az * * nem törhető szóközt ismét azért kellett alkalmazni, hogy ne csak a *toolkit* első szavát jelenítsék meg a böngészők.)

A Firefox és az Internet Explorer dőlt betűvel hívja fel a figyelmet, hogy tartozik meghatározás is a szóhoz. Ha azt kívánjuk, hogy valamennyi böngésző egyformán jelenítse meg, ill. hívja fel a figyelmet, CSS-el hozandó a kívánt formára a definíció.
(Lásd CSS rész Betűtípusok és Szöveg fejezeteit)

g) **Dőltbetűs** megjelenítést hoz létre az *……* páros címke (i=*italic*=dőlt betű). Műszaki kifejezések, idegen nyelvből vett idiómák, hajók neve, valaki gondolatai jelölhetők így. Az előző mondatban a latin kifejezést dőltbetűsen megjelenítve:

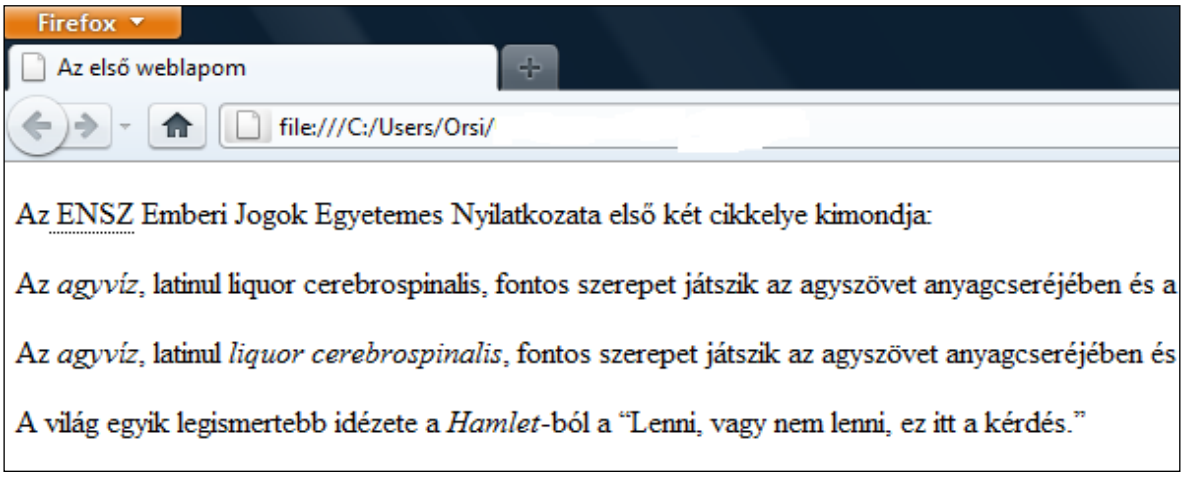
```
<p>Az<dfn title="az agyat és a gerincvelőt
    borító hártyák közötti résben 
   áramló folyadék"> agyvíz</dfn>, latinul <i>liquor
   cerebrospinalis</i>, fontos szerepet játszik az agyszövet
   anyagcseréjében és a kórokozók elleni védelemben.
</p>
```

h) A **címhivatkozás** páros címkéje a *<cite>……..</cite>* (*citation*=címhivatkozás). Nem szerzőre, hanem műre – könyvre, újságra, versre, dalra, színdarabra, másik forrásra – lehet így hivatkozni. Példánkban a Hamlet-re (mint drámára és nem mint személyre) hivatkozunk:

```
<p>A világ egyik legismertebb idézete a <cite>Hamlet</cite>-
   ból a <q>Lenni, vagy nem lenni, ez itt a kérdés.</q></p>
```

A böngészők dőlt betűvel jelenítik meg a címhivatkozást, ha változtatni akarunk ezen, az CSS-el történik *(lásd CSS rész Betűtípusok és Szöveg fejezeteit).*

Az e)-f)-g)-h) címkék megjelenítése:



i) Szövegben **kulcsszó**, katalógusban **terméknév** kiemelését szolgálja a *……* páros címke (b = *bold* = félkövér betű).

Pl. ha „A digitális videokamerák egyre növekvő hányadban a Full HD felbontást alkalmazzák." mondatban a videokamera a kulcsszó, a kódolás az alábbi:

```
<p>A digitális <b>videokamerák</b> egyre növekvő hányadban a
Full HD felbontást alkalmazzák.</p>
```

A böngészők alapértelmezetten félkövér betűkkel jelenítik meg, ha másképp akarjuk, CSS-el alakítható ki a kívánt formázás. *(lásd CSS rész Betűtípusok és Szöveg fejezeteit)*

j)-k)-l) **Tartalmi kiemelésre** a *……..* (*strong* = erős kiemelés), **kihangsúlyozásra** az *……..* (em = *emphasis* = hangsúly), **szövegben való kiemelésre** a *<mark>……..</mark>* (*mark* = jelzés, jel) páros címkék szolgálnak. A böngészők a *strong* esetében félkövér betűkkel, az *em* esetén dőltbetűvel, a *mark* használatakor sárga háttérrel jelenítik meg a megjelölt szavakat. Ha változtatni akarunk rajta, az CSS-el végezhető el.

Pl.: „A medencébe ugrálni veszélyes és szigorúan tilos !" mondatban a „szigorúan tilos"-t kiemelve:

```
<p>A medencébe ugrálni veszélyes és <strong>szigorúan tilos!
   </strong></p>
```

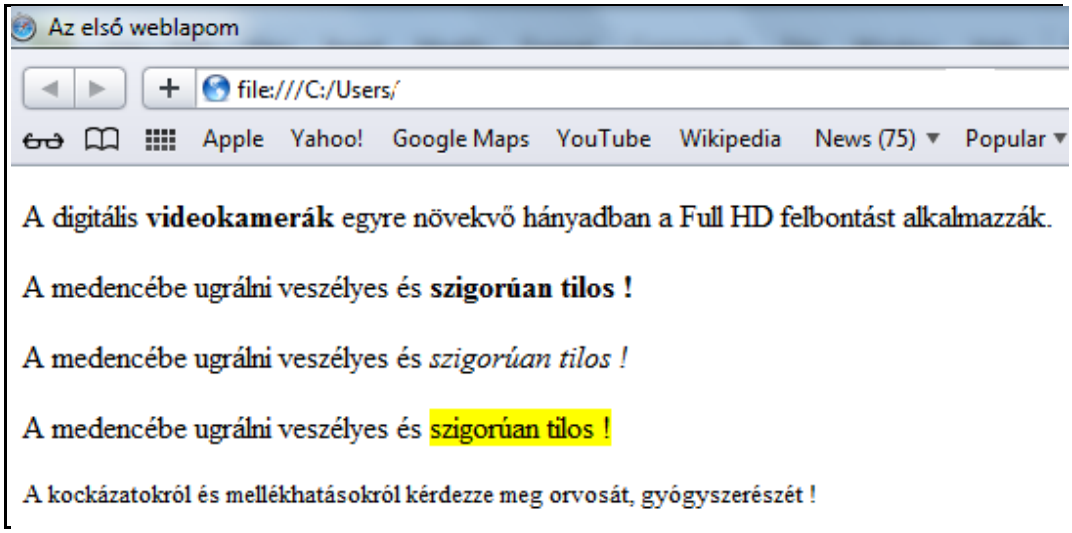
Az *em* vagy *mark* címkével a fentivel analóg módon történik a kódolás.

Figyelem: Pusztán formai meggondolásból a félkövér, dőltbetűs vagy színes háttérrel való meg-jelenítést CSS-el kell létrehozni, a fenti *i, b, strong, em, mark* HTML címkék tartalmi meg-különböztetést (is) jelölnek !

m) A *<small>……..</small>* páros címkével (*small* = kicsi) **kisbetűs** megjegyzéseket, szerzői jogra való korlátozásokat, stb. lehet blokkszinten bevinni. Pl. egy gyógyszerhirdetésekben ezerszer szereplő mondat:

```
<small>A kockázatokról és mellékhatásokról kérdezze meg
      orvosát, gyógyszerészét !</small>
```

Az i)-j)-k)-l)-m) címkék megjelenítése:



n) Az *<address>……</address>* páros címke a weblap szerzőjének vagy a hivatkozott információnak az **elérhetőségét** definiálja. Ha a weboldal törzs (body) részébe van írva, akkor a weboldal szerzőjének, ha a törzs egy szakaszába, akkor az adott szakasz szerzőjének az elérhetőségére vonatkozik. Blokkszintű elem, alapértelmezetten dőlt betűvel jelenik meg.

Figyelem! Általában egy postai címet nem az *<address>* címkével, hanem a *<p>* és *
* címkékkel lehet és kell kódolni!

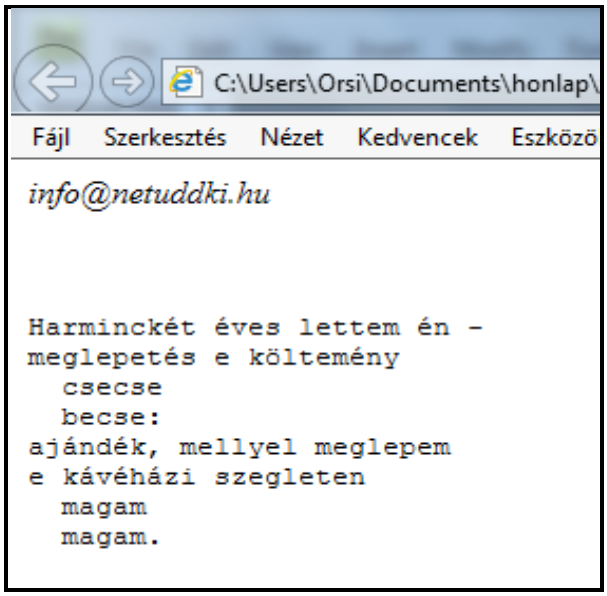
o) **Előre formázott szöveget** a *<pre>…...…</pre>* páros címkével (pre=*preformatted* = előre formázott) lehet a HTML lapba kódolni.

Mint már a korábbiakból kiderült, a HTML az elválasztó karakter (*inter-element whitespace*) típusától (szóköz, tabulátor, új sor) és számától függetlenül mindig csak egy szóközt használ. Az editorban szerkesztett un. ASCII-rajzok (karakterekből komponált képek), versek, stb. formázását viszont a *<pre>…... </pre>* címkék alkalmazásával a kódolás is megőrzi.

Pl. József Attila „Születésnapomra” c. versének első két versszakát az eredeti formában leírva, majd az előformázási kóddal körbefogva, a böngészők megőrzik az eredeti formát:

```
<pre>
Harminckét éves lettem én -
meglepetés e költemény
  csecse
  becse:
ajándék, mellyel meglepem
e kávéházi szegleten
  magam
  magam.
</pre>
```

Az n)-o) címkék megjelenítése:



p)-q)-r) **Szöveg változtatása** az eredeti tartalom megőrzésével két módon történhet:

- Mikor az eredeti és a módosított - beszúrt vagy törölt – szerkesztett tartalmat is egyaránt célszerű feltüntetni, az *<ins>……</ins>* (ins = *insert* = beilleszt, beszúr) és *~~……~~* (*delete* = töröl) páros címkék alkalmazandók.

Pl. egy utazásra való készülődéskor a „teendők” listájában az elvégzett feladatok és az időközben felmerült új teendők regisztrálása – az eredeti lista tartalmát is megőrizve – az alábbi listákban kódolható:

Az eredeti „teendők listája”:

```
<ul>
    <li>napolajat venni</li>
    <li>becsomagolni</li>
</ul>
```

A módosított „teendők listája”:

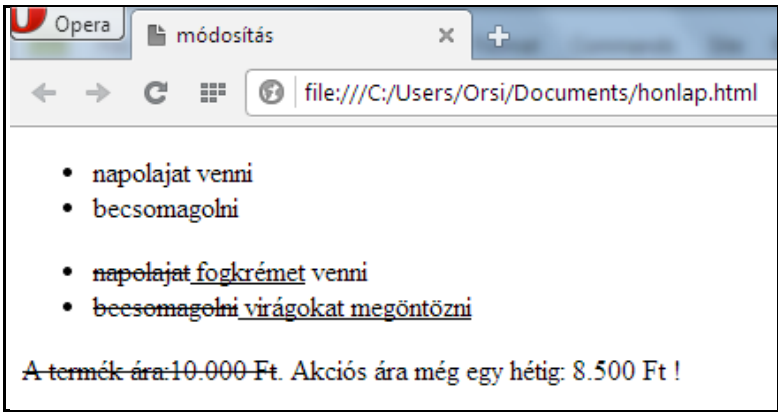
```
<ul>
   <li><del>napolajat</del><ins> fogkrémet</ins> venni</li>
   <li><del>becsomagolni</del><ins> virágokat megöntözni</ins>
   </li>
</ul>
```

A módosítás során a napolaj fogkrémre változott, a becsomagolás törölve lett, és a teendők a virágok megöntözésével lettek kiegészítve. (Mint látható, egy listaelemben több *ins* és *del* is elhelyezhető.)

A módosításokhoz a **datetime** jellemzővel hozzárendelhetők a beillesztés vagy törlés időpontjai – azonban a böngészők jelenleg ezt nem veszik figyelembe.

- Az *<s>……</s>* (s= *struck* = áthúzott) páros címkével azt jelöljük, hogy az eredeti információ jelenleg nem érvényes. Pl. árváltozás esetén:

```
<p><s>A termék ára:10.000 Ft</s>, Akciós ára még egy hétig:
   8.500 Ft !</p>
```



A törölt és nem aktuális/érvényes elemeket áthúzva, a beszúrt elemeket aláhúzva jelenítik meg a böngészők (ez konvenció, de CSS-el módosítható). (*lásd CSS rész Betűtípusok és Szöveg fejezeteit*)

s) Egy adott tartományon belüli érték definiálható a *<meter>……</meter>* páros címkével. Miután csak egy definiált értéktartományon belül értelmezett, mindig meg kell adni a tartomány két szélső értékét a *min* és *max* jellemzőkkel. Az opcionális *title* jellemző *tooltip* formájában abszolút számként vagy százalékosan mutatja meg a konkrét értéket, ha a kurzorral a tartomány fölé állunk (kattintani nem kell). Pl:

```
<p>Ekkorát ugrott a hármasugró az ugrógödörben:<br>
   <meter min="0" max="20" value="15" title="15 méter">
   </meter>
</p>
```

Megjegyzés: Az Internet Explorer-ek nem értelmezi a *meter* címkét.

t) Hosszú szavakba sortörési lehetőséget lehet (akár több helyen) is bekódolni. Ha a szó nem fér el az adott sorban, elválasztójel nélkül a megadott hely(ek)en töri meg a böngésző, és új sorban folytatja. Címkéje a *<wbr>* (wbr = *word break* = szó megszakítása), mely páratlan címke (üres elem). Például:

```
<p>Észak-Wales-ben van egy kis falu, amit általában Llanfair
  PG néven említenek, de a teljes neve egy 58 betűs szóból
  áll, és különböző rekordok könyve szerint ez a leghosszabb
  helységnév Nagy-Britanniában ( vagy egész Európában ? ).
  Turistabuszok özönlenek ide, hogy a látogatók trikót vagy
  bögrét vehessenek a falu teljes nevével, mely így hangzik:
  Llanfairpwllgwyngyllgogerychwyrndrobwll<wbr>llantysilio<wbr>
  gogogoch.
</p>
```

A böngésző ablaka szélességének fokozatos csökkentésével megfigyelhető, hogy a sztörés a kódolt helye(ke)n következik be.

Megjegyzés: Az Internet Explorer-ek nem értelmezi a *wbr* címkét.

u) A képletekben, függvényekben szereplő változók a *<var>……</var>* (var = *variable* = változó) páros címkével kódolhatók. A változókat a böngészők dőlt betűvel jelenítik meg. Például:

```
<p>Püthagorasz képletbe foglalta a derékszögű háromszög
   <var>a</var> és <var>b</var> befogói, ill. <var>c</var>
   átfogója mérete közötti törvényszerűséget.</p>
```

v) Számítógép billentyűzet karaktert jelöl a *<kbd>……</kbd>* (kbd = *keyboard* = billentyűzet) páros címke. Például:

```
<p>Böngészéskor a megnyitott oldal tartalmát az <kbd>F5</kbd>
   billentyű megnyomásával lehet frissíteni.</p>
<p>A <kbd><kbd>CTR</kbd>+<kbd>C</kbd></kbd>
   billentyűkombinációval lehet másolni.</p>
```

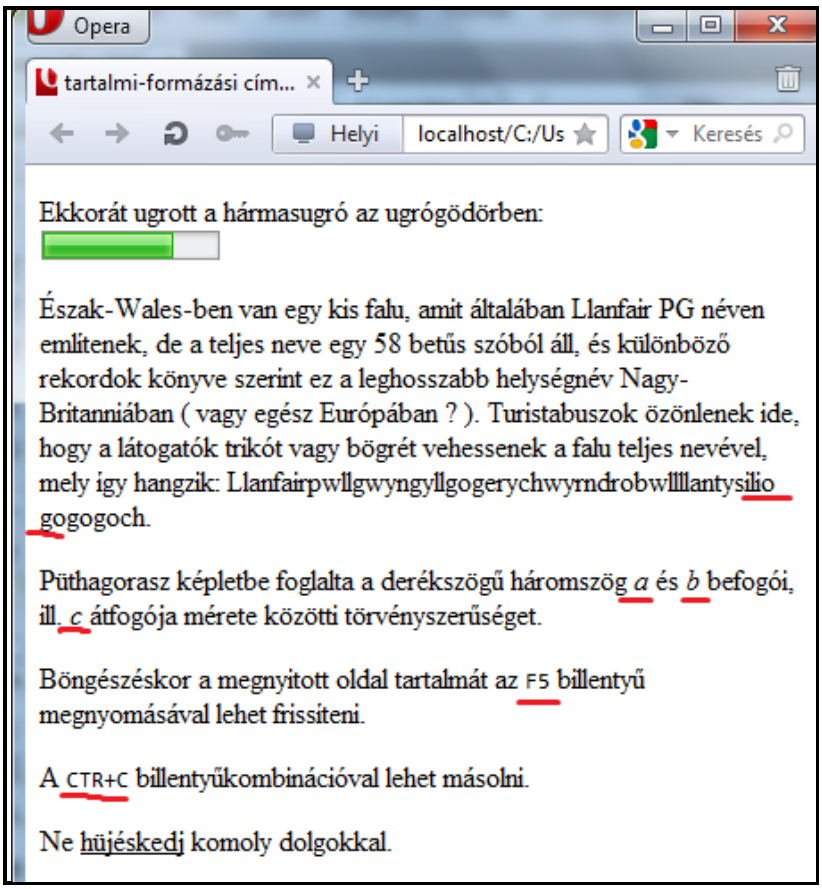
A második példában a dupla *<kbd></kbd>* nem sajtóhiba: a belső *kbd* elemek képviselik az egyes billentyűzet karaktereket, a külső *kbd* elem pedig a billentyűzet kombinációt. A szabvány helyesnek fogadja el azonban a külső *<kbd>……</kbd>* címke elhagyása esetén is a kódolást (mert logikailag enélkül is kikövetkeztethető, hogy billentyű kombinációról van szó).

w) Az *<u>….</u>* páros címkével (u = *underlined*=aláhúzva) jelölhető a kommentárt, értelmező megjegyzést igénylő szövegrészlet – pl. jelölve, hogy helytelenül (helyesírási vagy nyelvtani hibával) van írva. Példa:

```
<p>Ne <u>hüjéskedj</u> komoly dolgokkal.</p>
```

Ahol összekeverhető a (hagyományosan aláhúzással jelölt) hivatkozással, kerülendő a használata.

A s)-t)-u)-v)-w) címkék megjelenítése:



A felhasznált tartalmi/formázási HTML címkék:

Címke:	Funkciója:	
<sub>	alsó indexet definiál	páros címke
<sup>	felső indexet definiál	páros címke
<q>	rövidebb sorközi (inline) idézetet definiál	páros címke
<blockquote>	idézetblokkot definiál	páros címke
<abbr>	rövidítést, mozaikszót definiál	páros címke
<dfn>	meghatározást definiál	páros címke
<i>	dőltbetűs szöveget definiál	páros címke
<cite>	címhivatkozást definiál	páros címke
<meter>	egy ismert tartományon belüli értéket definiál	páros címke
<var>	változó értéket definiál	páros címke
<wbr>	szótörési lehetőséget definiál	páratlan címke
<b>	félkövér szöveget definiál	páros címke
	erős kiemelést definiál	páros címke
	kihangsúlyozást definiál	páros címke
<mark>	szövegrész kiemelését definiálja	páros címke
<small>	kisbetűs szöveget definiál	páros címke
<pre>	előformázott szöveget definiál	páros címke
<ins>	beszúrt szöveget definiál	páros címke
<del>	törölt szöveget definiál	páros címke
<s>	nem érvényes/már nem aktuális szöveget definiál	páros címke
<address>	dokumentum szerzőjének elérhetőségét definiálja	páros címke
<kbd>	billentyűzet karaktert definiál	páros címke
<u>	kommentárt igénylő, aláhúzott szövegrészt definiál	páros címke

2.12. A weboldal egyes részeinek formázását elősegítő HTML címkék

Ha a kódolt dokumentumban egy adott részénél nincsen olyan, a korábbi fejezetekben tárgyalt HTML címke, amihez formázás vagy hivatkozás lenne rendelhető, mesterségesen bevihetők ilyen címkék. Ha egy kijelölt résznek saját formázási környezet létrehozása szükséges, az ugyancsak a formázást elősegítő címkékkel hozható létre.

Ebben a fejezetben csak a formázást és hivatkozást elősegítő HTML címkék kódolását és azok alkalmazhatóságát tárgyaljuk, magát a részletes formázási technikát a CSS rész tartalmazza - itt csak egy-egy megjelenítési példa szerepel illusztrációként.

2.12.1. A * ……...… * páros címke (*division* = szakasz) egy **általános tárolóelem**et (általában a weblap egy logikailag összetartozó részét alkotó tömböt) jelöl ki. A benne elhelyezkedő tetszőleges tartalmakra (szöveg, lista, táblázat, kép, hivatkozások, stb.) egységesen lehet hivatkozni, ill. formázást hozzárendelni. A * * -ekkel olyan blokkszintű tömbök hozhatók létre, melyek a CSS részben ismertetett módon a weblapon változatos módon pozícionálhatók, és a weboldalak szerkezeti kialakítását szolgáló általános eszközként szolgálnak.

Ha több *div* szakasz is van egy weboldalon (és ez az általános eset), az egyes tárolóelemek az *id* jellemzőjük (*identifier*=azonosító) értékével (célszerűen a tárolóelemre jellemző névvel) azonosíthatók.

Az *id*-k értékeinek egyedieknek kell lenniük, azaz mindegyik csak egyszer szerepelhet egy weblapon, különben nem lesz egyértelmű az azonosításuk. Az *id*-knek betűvel kell kezdődniük, nem tartalmazhatnak szóközt, kisbetű-nagybetű érzékenyek, magyar megnevezés esetén célszerű ékezetmentes kisbetűs szavakat választani.

a) Példaként helyezzük el a weboldalon

- az első élelmiszeres listánkat, melyhez *Élelmiszerek listája* címsort rendelünk hozzá úgy, hogy az a listával egy egységet alkosson, együtt legyenek mozgathatók, hivatkozhatók

- és a rózsás képet, melyhez hasonló célból és módon hozzárendeljük a *Vörös rózsák* címsort.

A *div* elemek kódolása a listából ill. képből, és a címsorokból áll. A listás tárolóelem *id*-je (azonosítója) legyen „etel”, a rózsásé „rozsa” (az ékezetmentesség miatt).

Ha felül lesz a cím:

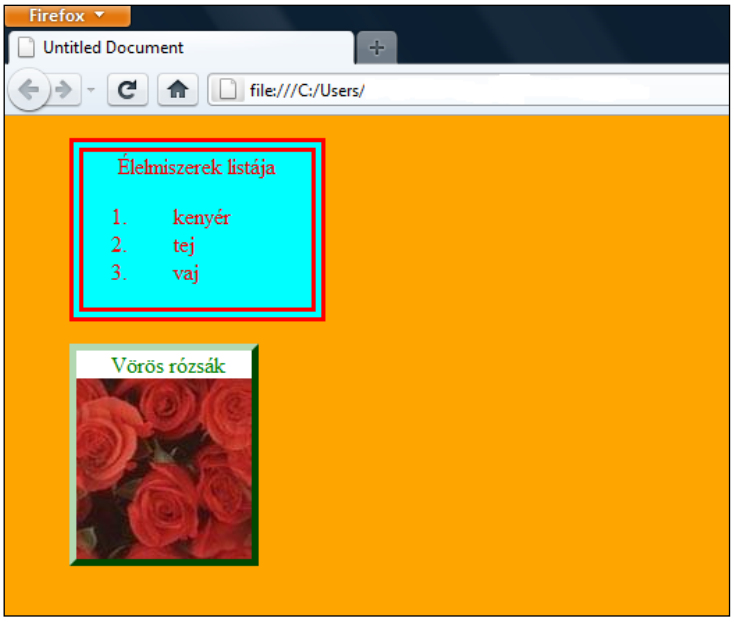
```
<div id="etel">
    <h1>Élelmiszerek listája</h1>
    <ol>
        <li>kenyér</li>
        <li>tej</li>
        <li>vaj</li>
    </ol>
</div>
<div id="rozsa">
     <h1>Vörös rózsák</h1>
     <img src="piros.jpg">
</div>
```

Ha alul lesz a cím:

```
<div id="etel">
    <ol>
        <li>kenyér</li>
        <li>tej</li>
        <li>vaj</li>
    </ol>
    <h1>Élelmiszerek listája</h1>
</div>
<div id="rozsa">
    <img src="piros.jpg">
    <h1>Vörös rózsák</h1>
</div>
```

Amíg a formázásra nem adunk meg CSS-tulajdonságokat, a HTML dokumentum megtekintésekor semmilyen változást nem okoz a fenti szakaszok kialakítása. Ha viszont formázzuk a *div* szakaszokat, számtalan megjelenítési módot (háttér, keret, betűszín, stb.) rendelhetünk hozzájuk, melyek a CSS-ben bármikor megváltoztathatók.

Például:



Figyelem! *Egy* blokkszintű elemet önmagában semmi értelme *div* blokkba rakni, hiszen az anélkül is *id* azonosítóval hivatkozhatóvá, formázhatóvá tehető.

Az *id* azonosító ugyanis általános jellemző, szinte minden címkéhez használható (kivételek a *head*, *html*, *meta*, *style* és *title*, de ezeknél értelmetlen is lenne a használata), tehát a címsorok, bekezdések, listák, táblázatok, idézetblokkok, stb. önmagukban is megkülönböztethetővé, hivatkozhatóvá és formázhatóvá tehetők:

```
<h2 id="…">…….....</h2>
<p id="……">…….....</p>
<ol id="…">…….....</ol>
<ul id="…">…….....</ul>
<dl id="…">…….....</dl>
<table id="……">…… </table>
<blockquote id="……">……</blockquote>
```

stb.

Nemcsak a blokkszintű, de a fenti néhány kivételtől eltekintve valamennyi elem ugyanígy kijelölhető és a továbbbiakban hivatkozható, formázható a hozzárendelt azonosítójával:

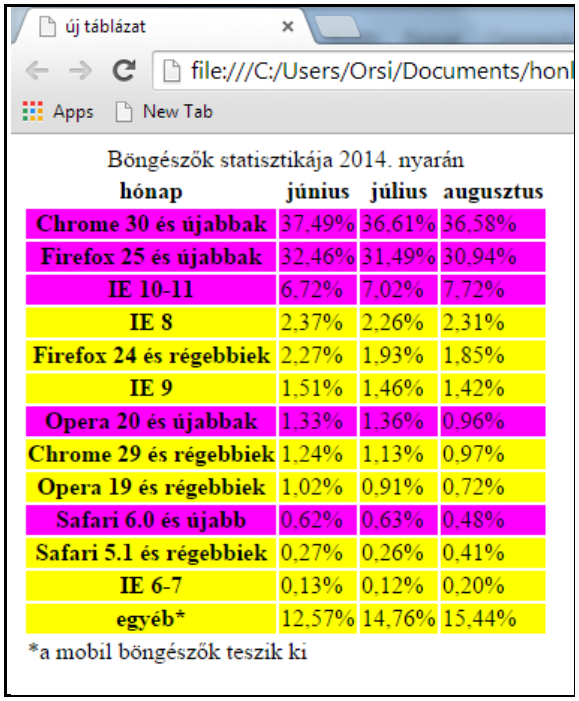
```
<th id="……">……</th>
<td id="……">……</td>
<li id="……">……</li>
<dt id="……">……</dt>
<dd id="……">……</dd>
```

stb.

Tetszőleges elemek *id*-kkel való hivatkozhatóvá és formázhatóvá tételének demonstrálására visszatérünk a 2014. nyári böngészőforgalmi táblázatunkhoz, amelyben a korszerűnek tekintett asztali böngészők adatait jelöljük ki különböző elemekhez rendelt *id*-kkel (aztán CSS-el háttérszínt rendelünk hozzájuk):

```
<table>
    <caption>Böngészők statisztikája 2014. nyarán</caption>
    <tr><th>hónap</th><th>június</th><th>július</th>
        <th>augusztus</th></tr>
```

```
    <tbody id="egy">
        <tr><th>Chrome 30 és újabbak</th><td>37,49%</td>
            <td>36,61%</td><td>36,58%</td></tr>
        <tr><th>Firefox 25 és újabbak</th><td>32,46%</td>
            <td>31,49%</td><td>30,94%</td></tr>
        <tr><th>IE 10-11</th><td>6,72%</td><td>7,02%</td>
            <td>7,72%</td></tr>
    </tbody>
    <tbody id="ketto">
        <tr><th>IE 8</th><td>2,37%</td>
            <td>2,26%</td><td>2,31%</td></tr>
        <tr><th>Firefox 24 és régebbiek</th><td>2,27%</td>
            <td>1,93%</td><td>1,85%</td></tr>
        <tr><th>IE 9</th><td>1,51%</td><td>1,46%</td>
            <td>1,42%</td></tr>
        <tr id="opera"><th>Opera 20 és újabbak</th>
            <td>1,33%</td><td>1,36%</td><td>0,96%</td></tr>
        <tr><th>Chrome 29 és régebbiek</th><td>1,24%</td>
            <td>1,13%</td><td>0,97%</td></tr>
        <tr><th>Opera 19 és régebbiek</th><td>1,02%</td>
             <td>0,91%</td><td>0,72%</td></tr>
        <tr><th id="safari">Safari 6.0 és újabb</th>
            <td id="junius">0,62%</td><td id="julius">
             0,63%</td><td id="augusztus">0,48%</td></tr>
        <tr><th>Safari 5.1 és régebbiek</th><td>0,27%</td>
            <td>0,26%</td><td>0,41%</td></tr>
        <tr><th>IE 6-7</th><td>0,13%</td>
             <td>0,12%</td><td>0,20%</td></tr>
        <tr><th>egyéb*</th><td>12,57%</td><td>14,76%</td>
             <td>15,44%</td></tr>
    </tbody>
        <tr><td colspan="4">*a mobil böngészők teszik ki</td>
        </tr>
</table>
```



Böngészők statisztikája 2014. nyarán

hónap	június	július	augusztus
Chrome 30 és újabbak	37,49%	36,61%	36,58%
Firefox 25 és újabbak	32,46%	31,49%	30,94%
IE 10-11	6,72%	7,02%	7,72%
IE 8	2,37%	2,26%	2,31%
Firefox 24 és régebbiek	2,27%	1,93%	1,85%
IE 9	1,51%	1,46%	1,42%
Opera 20 és újabbak	1,33%	1,36%	0,96%
Chrome 29 és régebbiek	1,24%	1,13%	0,97%
Opera 19 és régebbiek	1,02%	0,91%	0,72%
Safari 6.0 és újabb	0,62%	0,63%	0,48%
Safari 5.1 és régebbiek	0,27%	0,26%	0,41%
IE 6-7	0,13%	0,12%	0,20%
egyéb*	12,57%	14,76%	15,44%

*a mobil böngészők teszik ki

A fenti kódolás szöveges magyarázata:

- a táblázat törzsét két részre osztottuk - az első három sorra (*tbody id=”egy”*), mely összefüggően tartalmaz korszerű böngészősorokat, és a további részre (*tbody id=”ketto”*), melyben egymástól elkülönülve találhatóak meg az általunk kiválasztani szándékozott sorok

- a második (*tbody id=”ketto”*) törzsrészben lévő két sor közül az elsőt táblázatsorként (*tr id=”opera”*), a másodikat elemeiként, tehát táblázat fejlécként (*th id=”safari”*) és három táblázatcellaként (*td id="junius", td id="julius”* és *td id=”augusztus”*) jelöltük ki

- az *id*-k beiktatása nem okoz változást a megjelenítésben

- az *id*-khez való formázás (itt háttérszínek) hozzárendelését a CSS-rész tárgyalja

b) A weblap fő szerkezeti elemei *div* általános tárolóelemekben helyezhetők el.

Egy weblap törzsének (*body*) tipikus szerkezete a fejlécből (*header*), a befoglaló területből (*container* vagy *wrapper*) és a láblécből (*footer*) áll. A befoglaló terület tartalmazza a navigációs részt (*nav* = *navigation*) és a konkrét tartalmat (*content*). A tartalom további tömbökből állhat, mint pl. fő rész (*main*), szakasz (*section*), cikk (*article*), oldalsó információs sáv (*aside*), stb. Nem feltétlenül jelenik meg minden szerkezeti elem minden weblapon, és az egyes szerkezeti elemek a formázás kialakítása érdekében tetszőleges számú *div* részekre tovább oszthatók.

A fenti szerkezeti elemek kijelölése és azonosítása a *div* címkékkel és *id* jellemzőik értékeivel történhet:

- fejléc esetén (nem egyenlő a *head* = fejjel!)

```
<div id="header">
 fejléc tartalma ( pl. logo, embléma, jelmondat, stb. )
</div>
```

- befoglaló terület esetén

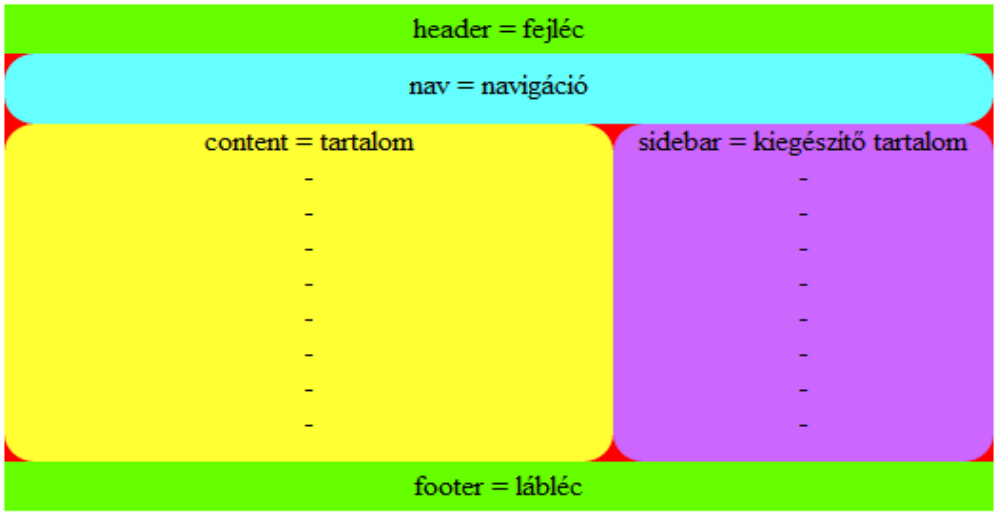
```
<div id="container"> vagy <div id="wrapper">
 navigációs rész, tartalom ( cikkekkel, oldalhasábos kiegészítésekkel, díszítő elemekkel, stb. )
</div>
```

- lábléc esetén

```
<div id="footer">
 lábléc tartalma ( pl. a weblap készítője, frissítés időpontja, copyright, csatlakozó dokumentu-
 mokhoz hivatkozások, szerző elérhetősége, stb. )
</div>
```

A fentiek analógiájára a navigációs résznek * …… *, a tartalomnak * …… *, azon belül * …… *, stb. módon alakítható ki saját tárolóelem (az egymás mellé helyezésüket majd a CSS végzi !).

A weboldal szerkezeti egységei *div* szakaszokkal kialakítva (pirossal a *container*):

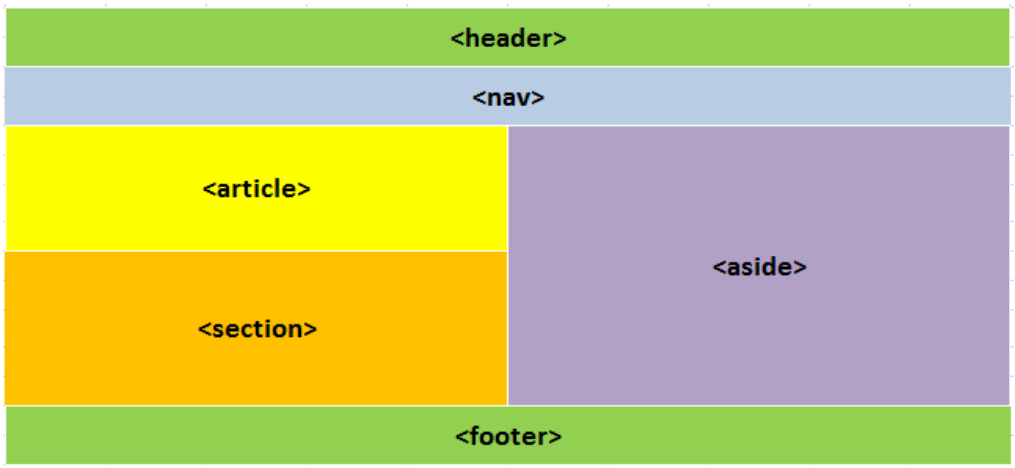


2.12.2. A főbb szerkezeti elemek saját, új HTML5 címkével is rendelkeznek:

`<div id="header">……</div>`	helyett	`<header>……</header>`
`<div id="nav">……</div>`	helyett	`<nav>……</nav>`
`<div id="footer">……</div>`	helyett	`<footer>……</footer>`

További, fő szerkezeti elemeket definiáló új páros címkék a *section* (weboldal egy – tipikusan címmel is rendelkező - szakasza), az *aside* (a fő gondolatmenethez érintőlegesen csatlakozó tartalom), és az *article* (újságcikk, blog-bejegyzés).

A weblap törzsének szerkezete tehát az új HTML címkékkel definiálva pl. az alábbi lehet (az egymás mellé helyezést a CSS végzi !):



A szerkezet HTML5 kódolása pedig:

```
<!doctype html>
<html lang="hu">
      <head>
            meta-adatok
      </head>
      <body>
            <header>
             fejléc tartalma
            </header>
```

```
            <nav>
                <ul><li>........</li>
                  navigációs rész(ek) tartalma
                  ( akár kettő is lehet, pl. a webhelyre és a
                  weblapra )
                <li>........</li></ul>
            </nav>
            <article>
             több is lehet belőle
            </article>
            <aside>
             több is lehet belőle
            </aside>
            <section>
             több is lehet belőle
            </section>
            <footer>
             lábléc tartalma
            </footer>
        </body>
</html>
```

A saját HTML címkés szerkezeti kialakítás nemcsak egyszerűbb a *div*-es kialakításnál, de a böngészők értelmezni is tudják, hogy milyen strukturális elemről van szó (a *div* címke jellemző/érték párosát nem értelmezik, csak egyszerűen megjelenítik, szemben a HTML5 jelentéstükröző címkéivel).

Megjegyzés: Az Internet Explorer 8 nem ismeri fel a 2.12.2. alatti HTML5 címkéket, csak a *div*-es megoldást jeleníti meg helyesen.

2.12.3. A *figure* tárolóelembe foglalhatók a képek, diagrammok, ábrák, listák, és a hozzájuk tartozó cím, felirat vagy magyarázat. Ezeket **a beágyazott elemeket** a *figure* (ábra) és *figcaption* (figure caption = ábra címe) páros HTML címkékkel lehet egy közös elemként kezelni és formázni. (A *figcaption* a *caption*-hez nagyon hasonló funkciót lát el.) A *div*-es szerkezetnél ez elegánsabb megoldás - például az első számozott listánkat „Élelmiszerek listája” címmel (lásd 2.12.1/a pont) ilyen módon egymáshoz rendelve a kódolás:

Ha felül van a cím:

```
<figure>
    <figcaption>Élelmiszerek listája</figcaption>
    <ol>
        <li>kenyér</li>
        <li>tej</li>
        <li>vaj</li>
    </ol>
</figure>
```

Ha alul van a cím:

```
<figure>
    <ol>
        <li>kenyér</li>
        <li>tej</li>
        <li>vaj</li>
    </ol>
    <figcaption>Élelmiszerek listája</figcaption>
</figure>
```

Vagy például egy képhez rendelt cím esetén a kódolás:

Ha felül van a cím:

```
<figure>
     <figcaption>Vörös rózsák</figcaption>
     <img src="piros.jpg">
</figure>
```

Ha alul van a cím:

```
<figure>
<img src="piros.jpg">
<figcaption>Vörös rózsák</figcaption>
</figure>
```

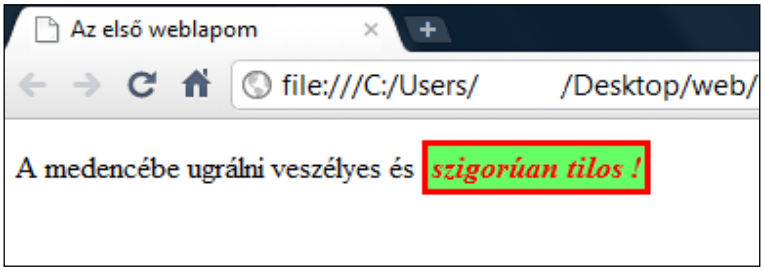
Megjegyzés: Az Internet Explorer 8 nem értelmezi a 2.12.3. alatti HTML5 címkéket, csak a *div*-es megoldást jeleníti meg helyesen.

2.12.4. **Soron belül** a *……..* páros címkével jelölhető ki olyan (*inline*) tartalom, ami külön formázható. A *span* címkét a *div*-hez hasonlóan az *id* egyedi azonosítóval lehet ellátni (ha egy oldalon több is van belőle).

Pl.: „A medencébe ugrálni veszélyes és szigorúan tilos!” mondatban a „szigorúan tilos”-t kiemelve – ezúttal csak formázás (nem mint korábban, tartalmi kiemelés) céljából:

```
<p>A medencébe ugrálni veszélyes és <span>szigorúan tilos !
   </span></p>
```

A *span* címke kódolásával önmagában nem történik változás a HTML oldal alapértelmezett megjelenítésében, a CSS-ben hátteret, betűszínt, betűstílust és szegélyt hozzárendelve viszont változik a megjelenítés:



2.12.5. Új funkciót vezet be a *<main>……..</main>* páros címke, mely a dokumentum törzs (*body*) fő tartalmi részét jelöli ki. Következésképpen csak egy lehet belőle egy dokumentumban, és az *article*, *aside*, *footer*, *header* vagy *nav* elemek nem tartalmazhatják. Az Internet Explorer-ek nem értelmezik a *main* címkét.

<u>A weboldal egyes szerkezeti részeinek formázását elősegítő HTML címkék:</u>

Cím:	Funkciója:	
	tömbszerű (blokkszintű) tároló (befoglaló) elemet definiál	páros címke
	soron belüli (inline) elemet definiál	páros címke
<header>	weboldal fejlécét definiálja	páros címke
<nav>	weboldal navigációs részét definiálja	páros címke
<footer>	weboldal láblécét definiálja	páros címke

<section> a tartalom egy tematikusan összetartozó részét definiálja páros címke
<aside> a fő gondolatmenethez érintőlegesen csatlakozó részt definiál páros címke
<article> egy section-on belül kialakított kisebb tartalmi elemet definiál páros címke
<main> dokumentum törzs fő tartalmi részét jelöli ki páros címke
<figure> és <figcaption> beágyazott elemnek és címének csoportját definiálják mindkettő páros címke

2.13. Hivatkozások

Webhelyen belüli vagy másik webhelyre történő ugrás, ill. külső erőforrások elérése formájában a hivatkozások adják a világhálós hipertérben valamilyen logikai összefüggés alapján történő barangolás lehetőségét.

Az *a* páros címke (a = *anchor* = horgony) a dokumentumon belül másik helyre, más dokumentumokra vagy másik webhelyekre mutató hivatkozás(oka)t, ill. weboldalon belüli hivatkozási célponto(ka)t rögzít („horgonyoz”) a weblapra. Hivatkozás kiindulópontjaként kötelező jellemzője a *href* (*hypertext reference* = hiperszöveges hivatkozás), melynek értéke a hivatkozás (ugrás) célpontja – általános alakja

```
<a href="……">……</a>
```

Oldalon belüli hivatkozási célpontjaként kötelező jellemzője egy *id* azonosító, melynek értéke az azonosító neve – általános alakja

```
<a id="……"></a>
```

Az egérmutató a hivatkozás kiindulópontja fölött kinyújtott mutatóujjú kézzé változik. A hivatkozásra kattintva a hivatkozott weboldal az aktuális oldal helyén jelenik meg.

2.13.1. Külső hivatkozások

a) Másik webhely főoldalára hivatkozás esetén a kódolás:

```
<a href="teljes URL-cím">hivatkozott webhely neve, vagy szó,
   fogalom és/vagy kép</a>
```

A *http://www.* használata a teljes URL-címben kötelező. A hivatkozott webhely gyorsabban nyílik meg, ha / jelet teszünk az URL végére.

A kódból alapértelmezés szerint csak a „hivatkozott webhely neve, vagy szó, fogalom és/vagy kép” jelenik meg. A szöveg aláhúzva, a betűk a még fel nem keresett hivatkozásoknál kék, a felkeresetteknél lila színnel láthatók. Egyes böngészők a hivatkozásként használt képet kék szegéllyel jelzik (a kék aláhúzás mintájára a szöveges hivatkozásnál). CSS segítségével ezek a beállítások megváltoztathatók.

Pl. az alábbi mondatot a weblapunkra beírva:

```
<p>Magyarországon a legjobban elterjedt kereső a
   <a href=http://www.google.hu>Google.</a></p>
```

A *Google* kék színnel és aláhúzva jelenik meg, ha ráállunk az egérrel, a kurzor kinyújtott mutatóujjú kézzé válik. Ha rákattintunk, a *Google Magyarország* kezdőlap nyílik meg, és visszatérve a saját weblapunkra ezután a *Google* szó lila színű marad.

b) Másik webhelyen belülre hivatkozás esetén a kódolás:

Ha a hivatkozott webhelyen belül egy konkrét weboldalra/mappára akarunk közvetlenül ugrani, a teljes URL-cím után /oldal vagy mappa nevet kell megadni.

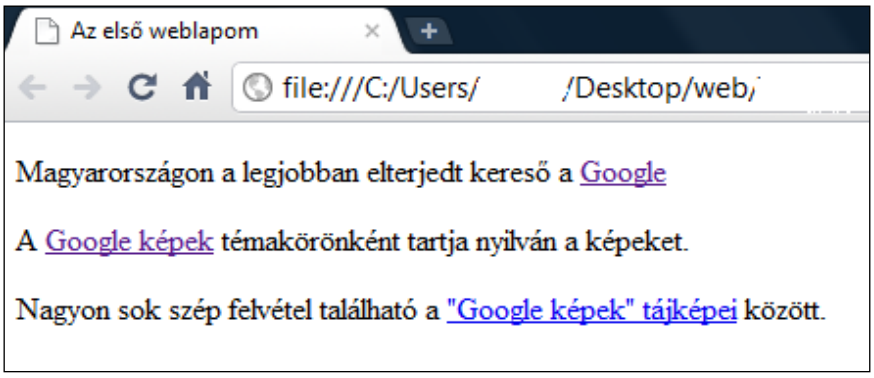
Pl. mélyebbre - a Google webhelyen közvetlenül a Google képekhez - hatolva a kódolás:

```
<p>A <a href="http://www.google.hu/imghp?hl=hu&tab=wi"> Google
   képek</a> témakörönként tartja nyilván a képeket.</p>
```

Még mélyebbre hatolva a webhelyben - pl. közvetlenül a tájképekhez ugorva - a kódolás:

```
<p>Nagyon sok szép felvétel található a <a href=
"http://www.google.hu/images?hl=hu&source=imghp&biw=1440&bih=
 807&q=tájképek&btnG=Képek+keresése&gbv=2&aq=f&aqi=&aql=&oq=
&gs_rfai=">"Google képek" tájképei</a> között.</p>
```

A már felkeresett *(visited)* és még fel nem keresett *(link)* hivatkozások alapértelmezett megjelenítése:



2.13.2. Belső hivatkozások

a) Azonos webhelyen belül másik weblapra hivatkozás esetén (ha a weboldalak ugyanabban a gyűjtőmappában vannak) a kódolás:

```
<a href="fájlnév.html">hivatkozott weboldal neve és/vagy kép,
szó fogalom</a>
```

Így lehet például a kezdőlapról a menü révén a webhely többi oldalára (ill. azokról vissza a kezdőlapra vagy egy további oldalra) ugrani. Alapértelmezetten a hivatkozott weboldal tetejére történik az ugrás.

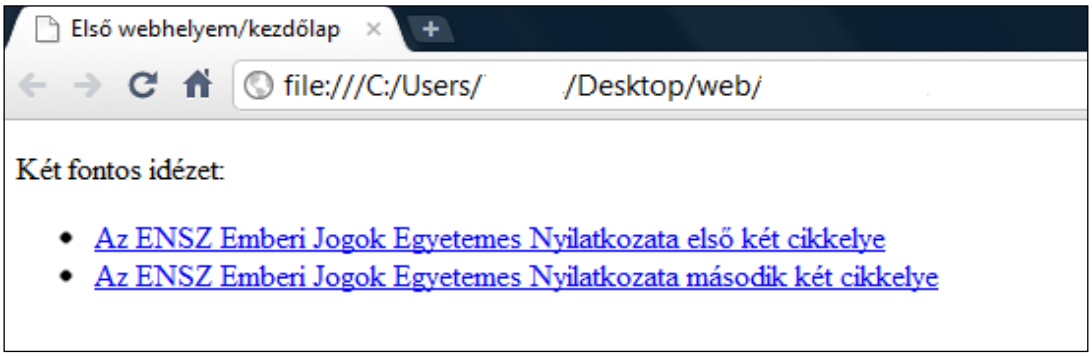
Az index.html kezdőlapon számozatlan lista elemeiként két hivatkozást felsorolva:

```
<!doctype html>
<html lang="hu">
     <head>
          <meta charset="utf-8">
          <title>Első webhelyem/kezdőlap</title>
          <link rel="stylesheet" href="display.css">
          <style>
          </style>
     </head>
```

```
    <body>
        <p>Két fontos idézet:</p>
        <ul>
            <li><a href="ensz1.html">Az ENSZ Emberi Jogok
                Egyetemes Nyilatkozata első két cikkelye
                </a></li>
            <li><a href="ensz2.html">Az ENSZ Emberi Jogok
                Egyetemes Nyilatkozata második két cikkelye
                </a></li>
        </ul>
    </body>
</html>
```

A hivatkozott oldalfájlok neveit a tartalmukra utalva célszerű megválasztani.

A dokumentumot *index.html*-ként a gyűjtőmappába mentve a webhely kezdőlapja így fog megjelenni a böngészőkkel:



A kék szín és aláhúzás jelzi, hogy hivatkozásokról van szó. Rájuk kattintva – amíg nem készülnek el a hivatkozott oldalak – még azt írja ki a böngésző, hogy a hivatkozott weblap nem található. Nyitni kell tehát két új fájlt, egyenként bekódolni a vonatkozó tartalmakat, majd a megfelelő neveken a gyűjtőmappába menteni őket.

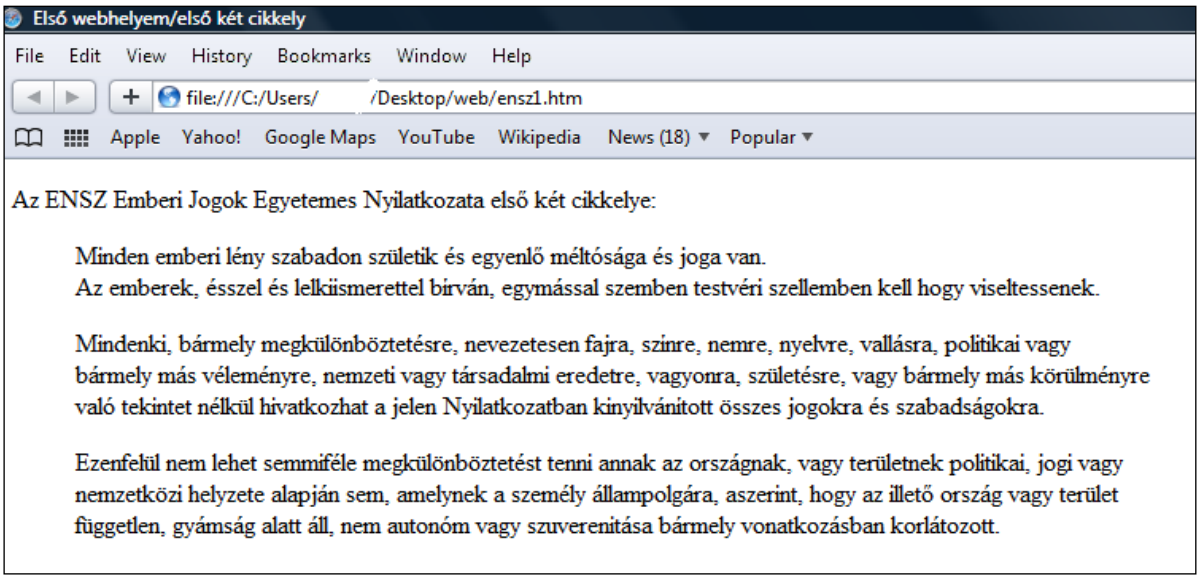
A két hivatkozott oldal kódolása:

1) az első két cikkelyt tartalmazó oldal (gyűjtőmappánkba *ensz1.html*-ként kell elmenteni):

```
<!doctype html>
<html lang="hu">
    <head>
        <meta charset="utf-8">
        <title>Első webhelyem/első két cikkely</title>
        <link rel="stylesheet" href="display.css">
        <style>
        </style>
    </head>
    <body>
        <p>Az ENSZ Emberi Jogok Egyetemes Nyilatkozata első
           két cikkelye:</p>
        <blockquote><p>Minden emberi lény szabadon
                    születik…….</p>
                <p>Mindenki, bármely megkülönböztetésre,
                    nevezetesen fajra, színre, nemre,
                    nyelvre,vallásra,…….</p>
```

```
                    <p>Ezenfelül nem lehet semmiféle
                       megkülönböztetést tenni annak az
                       országnak, vagy területnek politikai,
                       jogi vagy nemzetközi helyzete alapján
                       sem, amelynek a személy állampolgára,
                       aszerint, hogy az illető ország vagy
                       terület független, gyámság alatt áll,
                       nem autonóm vagy szuverenitása bármely
                       vonatkozásban korlátozott.</p>
            </blockquote>
        </body>
</html>
```

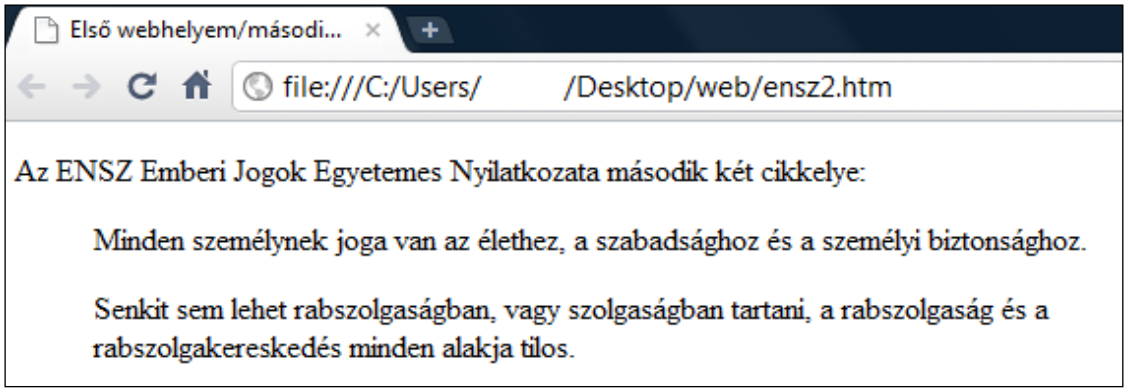
Önmagában az *ensz1* kódolt oldalt így mutatja a böngésző:



2) A második két cikkelyt tartalmazó oldal (gyűjtőmappánkban *ensz2.html*-ként kell elmenteni):

```
<!doctype html>
<html lang="hu">
    <head>
        <meta charset="utf-8">
        <title>Első webhelyem/második két cikkely</title>
        <link rel="stylesheet" href="display.css">
        <style></style>
    </head>
    <body>
        <p>Az ENSZ Emberi Jogok Egyetemes Nyilatkozata
           második két cikkelye:</p>
        <blockquote><p>Minden személynek joga van az élethez,
                       a szabadsághoz és a személyi
                       biztonsághoz.</p>
                    <p>Senkit sem lehet rabszolgaságban, vagy
                       szolgaságban tartani, a rabszolgaság
                       és a rabszolgakereskedés minden alakja
                       tilos.</p></blockquote>
    </body>
</html>
```

Önmagában az *ensz2* kódolt oldalt így mutatja a böngésző:

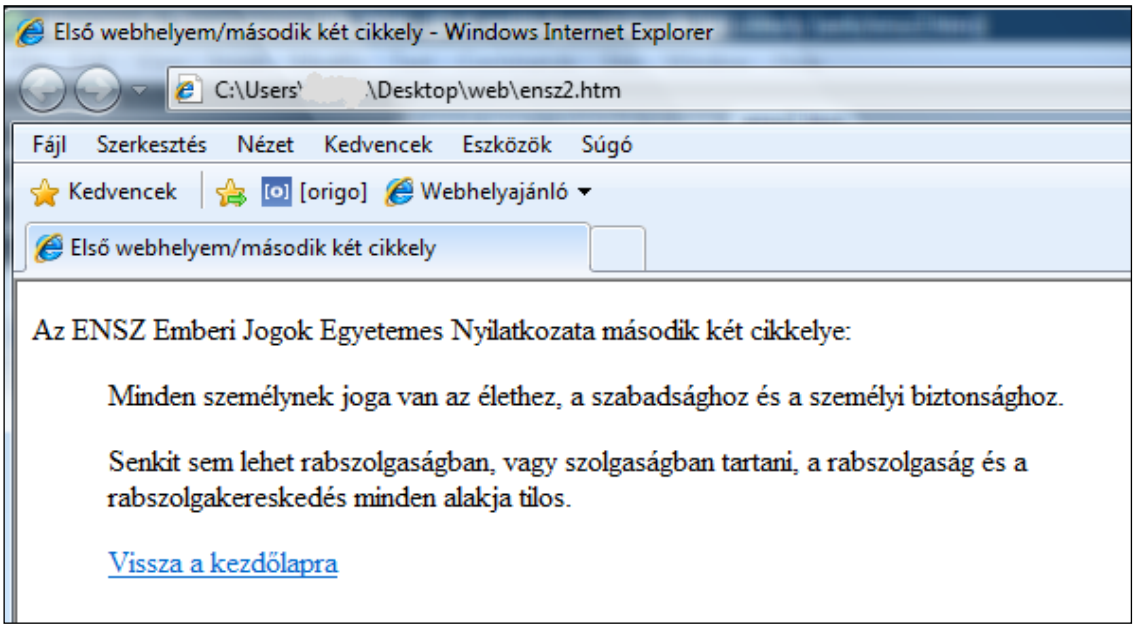


A kezdőlap listába foglalt hivatkozásaira kattintva most már a hivatkozott oldalra ugrik a böngésző. Ezzel az egyoldalas weblapokról továbblépve egy **több oldalas, komplett webhelyet** sikerült kialakítani. Természetesen minden oldal formázás nélkül, a böngészők alapértelmezett megjelenítésében látható, esztétikussá majd az oldalakhoz definiált stílussal alakíthatóak.

A hivatkozott (*ensz1* és *ensz2*) weboldalak tetejére vagy aljára a kezdőlapra visszaugrást is lehet kódolni:

```
<a href="index.html">Vissza a kezdőlapra</a>
```

Ekkor pl. az *ensz2* oldal így fog kinézni:



A *Vissza a kezdőlapra* hivatkozásra kattintva az *index.html* oldalra ugrik vissza a böngésző.

Természetesen bármelyik oldalról bármelyik másik oldalra való ugrás a fentivel analóg módon kódolható.

b) Ha nem a hivatkozott weboldal tetejére, hanem az oldalon belülre, egy kiválasztott részhez kívánunk ugrani, akkor a kódolás két lépésben történik:

b/1. A kiválasztott részhez (oldalon belüli ugrási célponthoz) egy egyedi azonosítóval ellátott horgonyt (*id* jellemzővel ellátott *a* címkét) kell kódolni:

```
<a id="..........."></a>
```

Az egyedi azonosítóval ellátott horgony a weblapon vizuálisan nem jelenik meg, és nem lehet rákattintani. Az azonosító ékezetmentes egyedi szó legyen. Az *>* horgony a célszó elé írandó.

b/2. Magát a hivatkozást (a hivatkozó weblapon, azaz a hivatkozott oldalon belüli kiválasztott részhez ugrást) az alábbi módon kell kódolni:

```
<a href="hivatkozott oldal fájlneve.html# horgony azonosítójának neve">hivatkozott weboldal neve, szó fogalom és/vagy kép</a>
```

Ennek nyilván csak hosszú hivatkozott oldalak esetén van értelme, egyébként az oldal egyben látszik a képernyőn és nem mutatkozik az oldalon belüli hivatkozás eredménye. Ezért pl. másoljuk ötször-hatszor egymás után az *ensz1* fájlba a két cikkely szövegét, és az utolsó másolat utolsó sorában jelöljük ki az *autonóm* szót mint ugrási célpontot. A hozzá rendelt horgony neve legyen pl. „*egy*”.

Ekkor az *index.html* kezdőoldalon a hivatkozások kódolása:

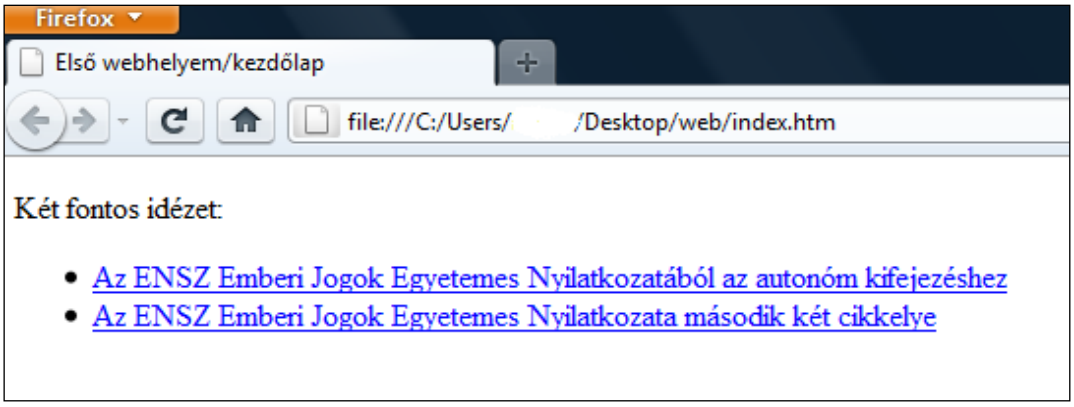
```
<ul>
    <li><a href="ensz1.html#egy">Az ENSZ Emberi Jogok
        Egyetemes Nyilatkozatából az autonóm
        kifejezéshez</a></li>
    <li><a href="ensz2.html">Az ENSZ Emberi Jogok
        Egyetemes Nyilatkozata második két cikkelye
        </a></li>
</ul>
```

Az *ensz1* fájlban a horgony kódolása a második cikkely utolsó másolatában:

```
<p>Ezenfelül nem lehet semmiféle megkülönböztetést tenni annak
 az országnak, vagy területnek politikai, jogi vagy nemzetközi
 helyzete alapján sem, amelynek a személy állampolgára,
 aszerint, hogy az illető ország vagy terület független,
 gyámság alatt áll, nem <a id="egy"></a>autonóm vagy
 szuverenitása bármely vonatkozásban korlátozott.</p>
```

Az ugrási célpontot CSS-el vizuálisan ki lehet emelni a környezetéből.

A módosított kezdőlap megjelenítése:



A módosított *ensz1oldal* ugrási célpontja:

Minden emberi lény szabadon születik és egyenlő méltósága és joga van.
Az emberek, ésszel és lelkiismerettel bírván, egymással szemben testvéri szellemben kell hogy viseltessenek.

Mindenki, bármely megkülönböztetésre, nevezetesen fajra, színre, nemre, nyelvre, vallásra, politikai vagy bármely más véleményre, nemzeti vagy társadalmi eredetre, vagyonra, születésre, vagy bármely más körülményre való tekintet nélkül hivatkozhat a jelen Nyilatkozatban kinyilvánított összes jogokra és szabadságokra.

Ezenfelül nem lehet semmiféle megkülönböztetést tenni annak az országnak, vagy területnek politikai, jogi vagy nemzetközi helyzete alapján sem, amelynek a személy állampolgára, aszerint, hogy az illető ország vagy terület független, gyámság alatt áll, nem **autonóm** vagy szuverenitása bármely vonatkozásban korlátozott.

c) Ugyanazon a weboldalon belüli ugrás a fentiekhez hasonló módon kódolható, csak elmarad az oldal neve és kiterjesztése. Pl. az oldal aljáról az oldal tetejére mutató link:

```
<a href="#eleje">Vissza az oldal elejére</a>        az oldal alján
<a id="eleje"></a>                                   az oldal elején
```

A weblap elejére mintegy tartalomjegyzékként is helyezhetők oldalon belüli hivatkozások, melyekre kattintva a weblap kívánt helyére lehet ugrani. Ha pl. *xyz* helyre vonatkozik a tartalomjegyzék egy pontja, akkor:

oldal tetején: `<a href="#xyz">xyz</a>`
az *xyz* előtt: `<a id="xyz"></a>`

Az *xyz* ugrási célpont CSS-el szintén vizuálisan kiemelhető.

2.13.3. Képek használata hivatkozásokhoz

Képek is használhatók mind külső mind belső hivatkozásokban szöveg helyett, ekkor az *a* horgony címkének magába kell foglalnia az *img* elemet, és ettől kezdve az *img* (maga a kép) lesz a hivatkozás azon része, amire rákattinthatunk:

```
<a href="fájlnév.html"><img src="képfájlnév.kiterjesztés"
                    width="……" height="…" title="…" alt"…"></a>
```

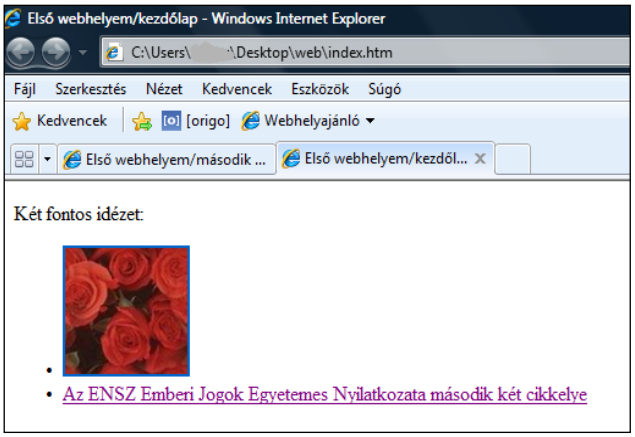
Pl.: a kezdő oldal hivatkozás menüjében az első hivatkozást a vörös rózsás képre cserélve, és a kép *title* és *alt* jellemzőiben megadva a hivatkozás tartalmát:

```
<a href="ensz1.html"><img src="piros.jpg" width="100px"
 height="100px" title="hivatkozás az ENSZ Emberi Jogok
 Egyetemes Nyilatkozata első két bekezdésére" alt=" hivatkozás
 az ENSZ Emberi Jogok Egyetemes Nyilatkozata első két
 bekezdésére"></a>
```

(Ez persze jelen esetben egy össze nem illő kombináció, csak az egyszerűség kedvéért került be a témához nem tartozó, de egyébként sokszor felhasznált kép.)

Ha képet használunk hivatkozásként, nem árt, ha valamilyen figyelemfelkeltő szöveg is van a képpel, hogy egyértelműen látszódjon róla, hogy hivatkozásról van szó. Az Internet Explorer-ek kék kerettel jelzik, hogy a kép hivatkozásként is használható, a többi böngésző ezt nem mutatja – ezért célszerű CSS-ben kék kerettel ellátni a képet. Az oldal esztétikus kialakítása szintén CSS útján történik.

Az alapértelmezett megjelenítés:



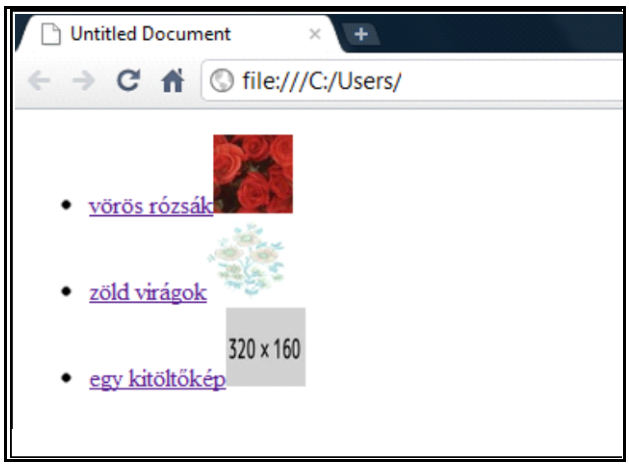
Képes hivatkozásokkal **(kép)galéria** (képtár) is készíthető. A menüben szöveggel és/vagy kicsinyített képekkel (u. n. bélyegképekkel) adjuk meg a hivatkozásokat (*thumbnail* = hüvelykujj körme, itt kis kép, bélyegkép), és az egyes képek (esetleg szöveges magyarázattal) a kívánt méretben és elrendezésben a hivatkozott oldalakon jelennek meg.

Pl.: egy kezdő oldal hivatkozás menüjében három szöveges hivatkozást és a hozzájuk tartozó bélyegképeket kódolva, majd három új fájlban az egyes képeket teljes méretben megjelenítve (ezek lesznek a *pirosnagy*, *zöldnagy*, *dummynagy html*-fájlok) az alábbi adódik:

Kezdő oldal a hivatkozásokkal:

```
<ul>
    <li><a href="pirosnagy.html">vörös rózsák</a><img
        src="piros.jpg" width="50px" height="50px">
    </li>
    <li><a href="zöldnagy.html">zöld virágok</a><img
        src="zöld.jpg" width="50px" height="50px">
    </li>
    <li><a href="dummynagy.html">egy kitöltőkép</a><img
        src="320x160.gif" width="50px" height="50px">
    </li>
</ul>
```

A kódolt kezdőoldal megjelenítése:



Az egyes hivatkozott oldalak kódolása:

vörös rózsák:

```
<h1>VÖRÖS RÓZSÁK</h1><br>
<img src="piros.jpg" width="300px" height="300px" alt="vörös
rózsák">
```

zöld virágok:

```
<h1>ZÖLD VIRÁGOK</h1><br>
<img src="zöld.jpg" width="300px" height="300px" alt="zöld
virágok">
```

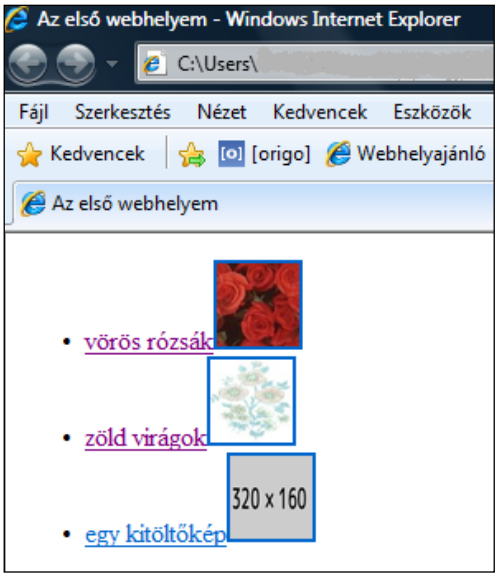
egy kitöltőkép:

```
<h1>KITÖLTŐKÉP</h1><br>
<img src="320x160.gif" alt="kitöltőkép">
```

Egy hivatkozott oldalt (pl. a vörös rózsását) így mutatnak a böngészők:

Ha azt akarjuk, hogy a szöveg és bélyegkép egyaránt hivatkozásként is használható legyen, akkor a kezdőoldal kódolása:

```
<ul>
     <li><a href="pirosnagy.html">vörös rózsák</a>
         <a href="pirosnagy.html"><img src="piros.jpg"
                  width="50px" height="50px"></a>
     </li>
     <li><a href="zöldnagy.html">zöld virágok</a>
         <a href="zöldnagy.html"><img src="zöld.jpg"
                  width="50px" height="50px"></a>
     </li>
     <li><a href="dummynagy.html">egy kitöltőkép</a>
         <a href="dummynagy.html"><img src="320x160.gif"
                  width="50px" height="50px"></a>
     </li>
</ul>
```



A képgalériák formázásával a CSS rész foglalkozik.

2.13.4. Hivatkozások használata multimédiához és e-mail-hez

Multimédia (mozgókép és hang) weboldalról történő **lejátszásá**nak egyik technikai megvalósítása hivatkozások útján lehetséges. Ebben az esetben az

```
<a href="……">multimédia fájl címe</a>
```

hivatkozással a kívánt fájl elérési útvonalát adjuk meg, és a böngészőre bízzuk, hogy a fájl kiterjesztése alapján megkeresse a lejátszáshoz szükséges módot bővítmény (*plug-in*) vagy külső segéd-alkalmazás (*helper application*) formájában.

Bővítmény használata esetén közvetlenül a böngészőablakban, külső segédalkalmazás igénybevételekor új ablakban történik a lejátszás. A fájl elérési útvonala közös gyűjtőmappa használata esetén egyszerűen a *fájlnév.kiterjesztés*.

E-mail írását lehet kezdeményezni a weboldalról a hivatkozás kódolásának alkalmazásával. Az

```
<a href="mailto:e-mail cím">SZÖVEG</a>
```

kódolásból a *SZÖVEG* látszik, ami tipikusan *Írjon nekünk*, vagy *E-mail*, vagy *Elérhetőségünk*, stb. Erre kattintva a látogató gépén megnyílik a levelezőprogramja, és az e-mail címzésében már az általunk kódolt *e-mail cím* szerepel.

Figyelem! A *SZÖVEG*-ben nem szerencsés megadni óvintézkedések (képi formátum, karakterkódolás, stb.) nélkül magát az e-mail címet, mert keresőrobotok *spam* (levélszemét, kéretlen levelek) címtár(ak)ba rakhatják azt be.

2.13.5. Hivatkozás használata letöltéshez

A hivatkozás általános alakját a *download* jellemzővel kiegészítve, azaz a

```
<a href="……" download>……</a>
```

kódolás esetén a hivatkozásra kattintáskor nem a megadott fájlra ugrás, hanem a megadott fájl letöltése következik be.

A *download* jellemző csak a *href* jellemzővel együtt használható. A böngésző automatikusan érzékeli a letöltendő fájl kiterjesztést és hozzáadja a fájlhoz (pl. *.html, .txt, .pdf,* stb.). Amennyiben nem adunk értéket a *download* jellemzőnek, akkor az eredeti fájlnévvel történik a letöltés, értékként viszont attól eltérő név is megadható, pl.:

```
<a href="http://www.……" download="konkurensek speckoi">A
   cégünket érdeklő műszaki adatok letöltése</a>
```

Megjegyzés: Az Internet Explorer-ek és Safari-k nem értelmezik és figyelmen kívül hagyják a *download* jellemzőt.

2.13.6. Képtérképek

A hivatkozások speciális alkalmazása a **képtérkép**, melyben egy kép egyes részeihez *(hotspot)* rendelhetők hivatkozások.

Hivatkozási területként háromféle geometriai alakzat (amelyik a legjobban illeszkedik a kijelölendő területre) helyezhető a kép egyes részeire: kör, derékszögű négyszög vagy általános sokszög. A hivatkozási területek kódja az <*area*> páratlan címke, kötelező jellemzői a *shape* (alak) és a *coords* (koordináták).

A **kör**t a középpontjával és a sugarával lehet definiálni, ennek megfelelően 3 számmal határozható meg: a kör középpontjának *x* és *y* koordinátája (a kép bal felső sarkától pixelben kifejezett távolsága) és a sugarának a nagysága (ugyancsak pixelben). A 3 adatot a fenti sorrendben értelmezi a böngésző, ha háromnál többet adunk meg, a többit egyszerűen nem veszi figyelembe. Egy kör alakú hivatkozási terület kódolása tehát:

```
<area shape="circle" coords="……,……,……" href="……" alt="……"
      title="……">
```

A **derékszögű négyszög**et (négyzet, téglalap) két sarokpontjának *x* és *y* koordinátáival, tehát 4 számmal lehet meghatározni. A böngésző az első két számot a négyszög bal felső sarkának a kép bal felső sarkától (pixelben) mért vízszintes és függőleges távolságaként, a második két számot pedig a négyszög jobb alsó sarkának koordinátájaként értelmezi – ha négynél több adatot adunk meg, a többit figyelmen kívül hagyja. Egy négyzet vagy téglalap alakú hivatkozási terület kódolása tehát:

```
<area shape="rect" coords="……,……,……,……" href="……"
      alt="……" title="……">
```

Az **általános sokszög**eket (háromszögek, nem derékszögű négyszögek, ötszögek, hatszögek, stb.) csúcspontjaik *x* és *y* koordinátáinak megadásával lehet definiálni, tehát minimum hat (és a továbbiakban páros számú) adatot kell vesszőkkel felsorolva megadni. Egy háromszög alakú hivatkozási terület kódolása tehát:

```
<area shape="poly" coords="……,……,……,……,……,……" href="……"
      alt="……" title="……">
```

A hivatkozási területek (*hotspot*-ok) száma nincsen sem fajtánként, sem összességében korlátozva. A hivatkozások mutathatnak az oldalon vagy webhelyen belüli célpontokra vagy más webhelyekre is, és egy célponthoz több <*area href*=”…….”> elem is tartozhat.

A fenti módon kialakított hivatkozási területeket egy *<map>….</map>* páros címkék alkotta (*map* = térkép) képtérkép elembe foglaljuk be, melynek valamilyen *id=”……”* egyedi azonosítót is kell adni, amellyel a képünkhöz tudjuk rendelni a képtérképet.

A képtérkép kódolása tehát:

```
<map id="……">
   <area shape="……" coords="……" title="……" alt="……" href="……">
   <area shape="……" coords="……" title="……" alt="……" href="……">
   <area shape="……" coords="……" title="……" alt="……" href="……">
                                  .
                                  :
   <area shape="……" coords="……" title="……" alt="……" href="……">
</map>
```

Ezt a képtérképet rendeljük hozzá magához a képhez az alábbi módon:

```
<img src="képfájl neve.kiterjesztése" alt="kép helyettesítő
szövege" title="kép címe" width= "szélesség" height="magasság"
usemap="#a map címkében megadott id azonosító">
```

Vagyis a képeknél szokásos jellemzők a *usemap*-el bővülnek, melynek értéke megegyezik a képtérkép (*<map>….</map>*) azonosítójával.

Vegyünk egy konkrét példát - adott egy terített asztal képe, melynek néhány ételéhez hivatkozásokat rendelünk:

```
<p>Egy tengeri halvacsora<img src="IMG_1937.jpg" alt="terített
   asztal" title="bouillabaisse" width="325" height="245"
   usemap="#halvacsora">Marseille-ben.</p>
```

A kép kódolásában az újdonság a *usemap* jellemző, mellyel a *halvacsora* azonosítójú képtérképet rendeljük majd hozzá. Ez egyelőre nem változtat semmit a szokásos képen:

A két legnagyobb burgonyát körökkel nagyjából lefedjük, és egy főtt burgonyás receptekkel foglalkozó, találomra kiválasztott webhelyre mutató külső hivatkozást kódolunk be:

```
<area shape="circle" coords="69,105,15" title="főtt burgonya"
   alt="főtt burgonya" href="http://www.burgonyareceptek.com/
   receptek/koretek/fottburgonya/fott_burgonya_receptek.html">
<area shape="circle" coords="46,68,16" title="főtt burgonya"
   alt="főtt burgonya" href="http://www.burgonyareceptek.com/
   receptek/koretek/fottburgonya/fott_burgonya_receptek.html">
```

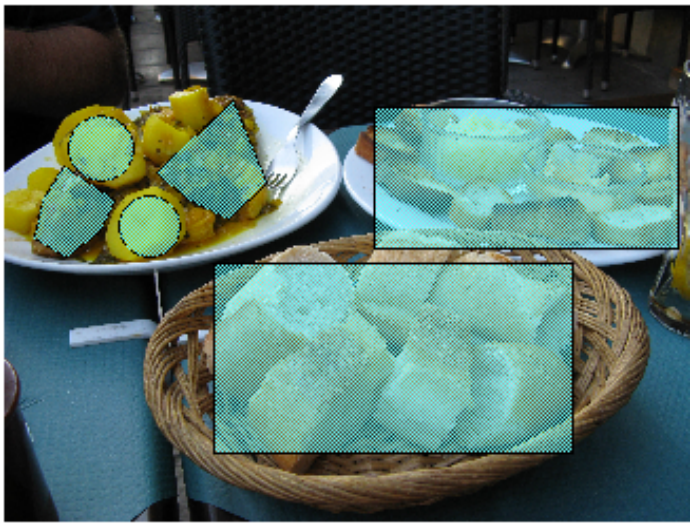
A kenyeres kosarat és tányért téglalapokkal nagyjából lefedjük, és egy-egy baguette-es ill. előételes receptekkel foglalkozó, találomra kiválasztott webhelyre mutató külső hivatkozást kódolunk be:

```
<area shape="rect" coords="99,122,270,213" title="baguette"
    alt="baguette" href="http://www.schef.hu/francia-baguette-
    recept">

<area shape="rect" coords="175,48,320,116" title="előétel"
    alt="előétel" href="http://www.nosalty.hu/receptek/fogas/
    hideg-eloetel">
```

A hal fejét és farkát sokszögekkel nagyjából lefedjük, és egy halreceptekkel foglalkozó, találomra kiválasztott webhelyre mutató külső hivatkozást kódolunk be:

```
<area shape="poly"
   coords="28,76,18,92,14,110,29,119,45,107,52,92" title="hal"
   alt="hal" href="http://www.haletel.lap.hu">

<area shape="poly" coords="108,45,124,84,106,101,86,91,72,79"
   title="hal" alt="hal" href="http://www.haletel.lap.hu">
```

A fenti kódolásokkal kialakított hivatkozási területek (a fennmaradó részek hivatkozás szempontjából inaktívak):

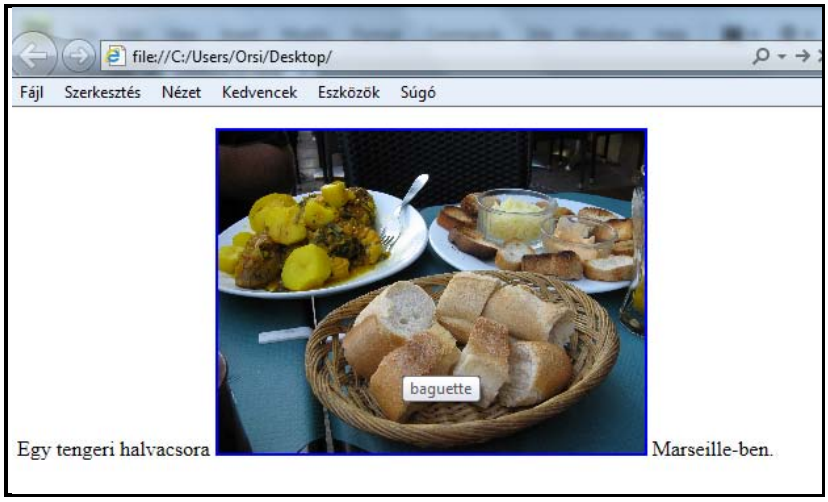


A 6 darab <*area*> elemet a **<*map id*=*"halvacsora"*>**….....…..**</*map*>** képtérkép tárolóelembe foglalva elkészült a képtérkép. A <*map*> tárolóelemnek nem kell közvetlenül az <*img*> elemet követnie a kódolásban.

Figyelem! Az *img* kép *width* vagy *height* méretének utólagos megváltoztatása esetén a hivatkozási területekhez képest – mivel azok a kép bal felső sarkához rögzített helyzetben maradnak - az alattuk lévő képrészletek (amire ráillesztettük őket) arréb csúsznak, így a képtérkép pontatlanná, nagyobb változtatás esetén használhatatlanná válik.

Ha a hivatkozási területekre visszük a kurzort, megjelenik az adott kattintható régiónak a neve és a kinyújtott mutatóujjú kéz (utóbbi a képen nem látszódik), mely jelzi a hivatkozhatóságot. Kattintáskor a megjelölt webhelyeket nyitja meg a böngésző. Az inaktív terü-

leteknél a normál képeknél szokásos módon buborékszövegben a kép címe (*title tooltip*) jelentkezik.



A hivatkozásoknál felhasznált HTML címkék:

Címke:	Funkciója:	
<a>	horgony, hivatkozásokat és célpontokat definiál	páros címke
<map>	(ügyféloldali) képtérképet definiál	páros címke
<area>	hivatkozások részére területet definiál	páratlan címke

2.14. Űrlapok

Az űrlapok a webhely látogatóival való kommunikációt teszik lehetővé, információadásnak vagy visszajelzéseknek az eszközei. A HTML5-nek - a multimédia mellett – az űrlapok a legtöbb újat hozó területe, ennek megfelelően egyes funkciók szabványosítása és böngészők általi támogatottsága még a megvalósítás stádiumában van.

Az űrlapok kialakítása HTML-el, formázásuk CSS-el történik, azonban működtetésükhöz szkript-nyelvek alkalmazása szükséges, mely meghaladja a „Szabványkövető statikus honlapok szerkesztése” kereteit - itt tehát „csak” az űrlapok létrehozásáról lesz szó.

Az űrlapot definiáló páros címke a *<form>……..</form>* (*form*=űrlap, űrlapelem). Hasonlóan a táblázat *table* vagy a listák *ol/ul/dl* címkéihez, önmagában a *form*-nak sincsen megjelenési formája, csak a tartalmi elemeivel együtt alkot egy űrlapot.

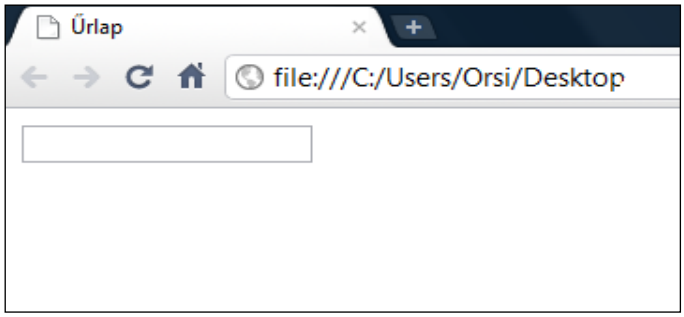
Az űrlap vezérlőelemei (*controls*) információbeviteli mezők vagy választást lehetővé tevő alakzatok. A többségük az *input* páratlan címkével (*input*=adatbevitel) adható meg, melynek *type* jellemzője (*type* = jelleg, fajta, típus) határozza meg az bevitt adat jellegét (néhány más elemnek pedig saját címkéje van).

2.14.1. Az *input* címke

Az űrlap kódolása a legegyszerűbb esetben:

```
<form>
    <input>
</form>
```

Alapértelmezetten az *input* egy egysoros szövegmezőt definiál. Megjelenítése:



A szövegmezőbe kattintva egy formázatlan szöveg írható be. Új sort nem lehet kezdeni, a mező hossza alapértelmezetten kb. 20 karakter – ennél hosszabb szöveg is beírható, ekkor balra vízszintesen fut a további beírás a szövegmezőben.

a) Kiegészítés a fenti alapesethez:

- az *input* elem egységes tárgyalása érdekében az egysoros szövegmezőnél is adjuk meg a *type* jellemzőt, melynek értéke ebben az esetben *text* (szöveg). Ez a *type* alapértelmezett értéke, tehát nem hibás, ha nem definiáljuk, de jobb megszokni, hogy a többféle *input*-hoz különböző *type* jellemzők tartoznak

- az űrlapon megadott adatok az elküldés után név-érték páronként értékelhetők ki, ezért mind az űrlapnak (a *form* elemnek), mind az adatbevitelt ill. választást lehetővé tevő űrlap-elemeknek *name* jellemzővel (a választógombok kivételével) egyedi értéket – azaz egyedi neveket - kell adni

- az egyes űrlapelemek *inline* alapértelmezettek – ha nem egy sorban kell elhelyezkedniük, célszerű * …… * bekezdésekbe kódolni őket.

Fentiek alapján tehát egy egysoros szövegbeviteli mezővel rendelkező űrlap kódolása:

```
<form name="gyakorlat">
      <p><input type="text" name="szoveg"></p>
</form>
```

A beírt jellemzők/értékek nem okoztak változást a megjelenítésben.

b) Az egysoros szövegmező mind tartalmilag, mind formailag különböző jellemzőkkel módosítható, ill. címfelirattal látható el:

- Tartalmi pontosítás (amit már a *type="text"*-nél is elvégeztünk):

type="tel" esetén telefonszámot
type="email" esetén e-mail címet
type="search" esetén keresőszót
type="url" esetén egy internetes címet
type="password" esetén jelszót tartalmaz a szövegmező.

Megjelenítésben csak a jelszómező tér el a többitől – a beírt karakterek helyett a böngésző pontokat vagy csillagokat mutat, ami megakadályozza, hogy illetéktelenek elolvashassák az érvényes jelszót. Az internetes címet abszolút URL-ként (*http://www*.....) kell megadni.

- Valamennyi fenti *type*-ra érvényes formai módosítások:

value=”…..” esetén a megadott kezdőérték (szöveg) jelenik meg a mezőben – a felhasználó bele- vagy hozzáírva, törölve módosíthatja, átírhatja az általa kívánt szövegre

size=”….” esetén a szövegmező hossza karakterszámmal megadható – ez a mező kisebb vagy nagyobb méretével orientálja a felhasználót, de nem korlátozza a bevihető karakterek számát - ennél hosszabb szöveg esetében vízszintesen fut balra a további beírás a szövegmezőben

maxlength=”…..” esetén a felhasználó által bevihető karakterek maximális száma korlátozható – ha ennél többet ír be, a plusz karakterek már nem jelennek meg (ez a karakterszám független a szövegmező méretétől, rövidebb és hosszabb is lehet annál)

readonly (ennek a jellemzőnek nem kell értéket adni) - csak olvasható a szövegmező, vagyis a felhasználó nem tud beleírni, ill. az esetleges kezdőértéket módosítani

hidden (ennek a jellemzőnek nem kell értéket adni) – rejtett (*hidden* = rejtett) mező, a felhasználó sem nem látja a szövegmezőt, sem megváltoztatni nem tudja a szöveget

autofocus az űrlap betöltődésekor a kurzor automatikusan a megadott szövegmező elejére áll, lehetővé téve az adott mezőbe az azonnali információbevitelt

```
<form name="gyakorlat">
     <p><input type="text" name="pelda1" value="kezdőszöveg"
                size="50" readonly></p>
     <p><input type="text" name="pelda2" value="kezdőszöveg"
                size="95" hidden></p>
</form>
```

- Szövegmezők elemfeliratozása:

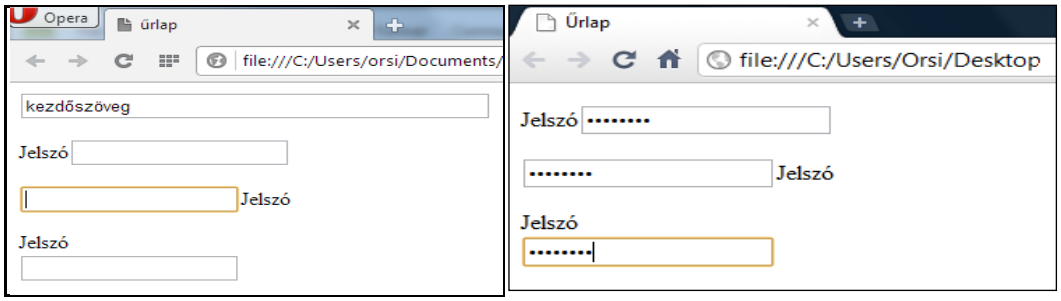
A *<label>……..</label>* páros címkével (*label*=elemfelirat, címfelirat) adható felirat az egyes szövegmezőkhöz. A *label* kezdő- és zárócímkéje közrefogja a felirattal ellátandó *input* elemet. A kódolás pl. jelszó mezőhöz, ha előtte, utána vagy fölötte helyezkedik el a felirat:

```
<form name="gyakorlat">
    <p><label>Jelszó<input type="password" name="jelszo1">
       </label></p>

    <p><label><input type="password" name="jelszo2" autofocus>
       Jelszó</label></p>

    <p><label>Jelszó<br><input type="password" name="jelszo3">
       </label></p>
</form>
```

Akár a szövegmezőbe, akár az elemfeliratra kattintunk, egyaránt a szövegmező elejére ugrik a kurzor, és megkezdődhet a szöveg beírása. Az *autofocus* jellemzővel is ellátott mező esetében nincsen kattintásra sem ehhez szükség, betöltődéskor automatikusan oda kerül a kurzor. Ha nem az *autofocus*-ban lévő elemmel akarjuk kezdeni az adatbevitelt, a kurzorral egyszerűen csak az általunk választott helyre kell ugrani.

A fenti kódolások alapértelmezett megjelenítésekor látszik, hogy:

- a rejtett mezőnek a helye sincsen kihagyva
- az fókuszt a Chrome és Opera a mező szegélyének színes bekeretezésével tovább hangsúlyozza



Megjegyzés: A felhasználó tájékoztatására szolgáló rövid szöveget a *value* jellemzőn kívül a *placeholder* jellemzővel is megadhatjuk (ekkor nem fekete, hanem szürke betűkkel jelenik meg a kezdőszöveg, egyébként úgy viselkedik, mint a *value*).

c) Választóelemek kialakítása:

c/1.) A *type="checkbox"* jellemző/érték esetén **jelölőnégyzetbe** történő bejelölés (kattintás) formájában előre megadott választási lehetőségek közül egy vagy több kiválasztása lehetséges.

Ha például az a kérdés, hogy milyen nyelven (magyar, angol, német és/vagy francia) tud a felhasználó szakirodalmat olvasni, akkor a jelölőnégyzetek kódolása - *value*-ként a választási lehetőségeket is definiálva - az alábbi:

```
<p>
  <label><input type="checkbox" name="nyelv1" value="magyar">
                 magyar</label>
  <label><input type="checkbox" name="nyelv2" value="angol">
                 angol</label>
  <label><input type="checkbox" name="nyelv3" value="nemet">
                 német</label>
  <label><input type="checkbox" name="nyelv4" value="francia">
                 francia</label>
</p>
```

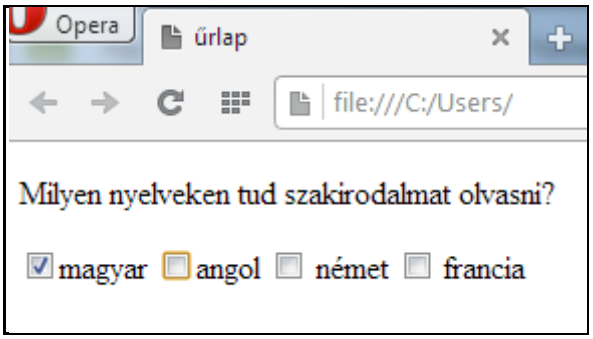
Tekintettel arra, hogy az egyes jelölőnégyzetek *inline* elemek, sorban egymás mellé kerültek. Az elemfeliratozás az egysoros szövegmezőnél leírtakkal azonos módon működik.

Ha valamelyik választási lehetőséget előre felkínáljuk, az adott jelölőnégyzethez a *checked* (érték nélkül használható) jellemzőt kell beírni - természetesen a felhasználó az előre felkínált gombra kattintva a jelölést bármikor törölheti.

Az *autofocus* jellemző az egyszerű szövegmezőnél leírtakkal megegyezően működik.

A kérdéssel is kiegészített kódolás és a megjelenítése:

```
<form name="gyakorlat">
<p>Milyen nyelveken tud szakirodalmat olvasni?</p>
<p><label><input type="checkbox" name="nyelv1" value="magyar"
                 checked>magyar</label>
  <label><input type="checkbox" name="nyelv2" value="angol"
                 autofocus>angol</label>
  <label><input type="checkbox" name="nyelv3" value="nemet">
                 német</label>
  <label><input type="checkbox" name="nyelv4" value="francia">
                 francia</label>
</p>
</form>
```



Az autofókuszban lévő elemet a Chrome és Opera színes szegéllyel, az Internet Explorer szaggatott szegélyű jelölőnégyzettel mutatja (a Firefox nem jelöli vizuálisan).

c/2.) A *type="radio"* jellemző/érték esetén **választógombra** (rádiógombra) kattintás révén a felkínált válaszok közül csak egynek a kijelölése lehetséges.

Ha például 3 lehetőség (igen, nem, talán) közül egyet lehet választani, akkor a választógombokra vonatkozó kódolás (*value*-ként a választási lehetőségeket definiálva):

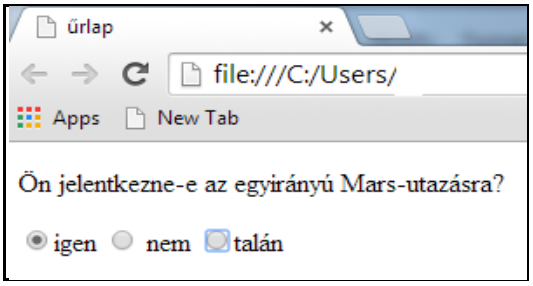
```
<p><label><input type="radio" name="valasztas" value="igen">
                 igen</label>
   <label><input type="radio" name="valasztas" value="nem">
                 nem</label>
   <label><input type="radio" name="valasztas" value="talan">
                 talán</label></p>
```

Mivel egy bekezdésbe kódoltuk az alapértelmezetten *inline* választógombokat, egymás mellett helyezkednek el. A feliratozás és autofókusz az egysoros szövegmezőnél leírtakkal azonos módon működik. Az egy kérdésre vonatkozó (ugyanabba a választógombcsoportba tartozó) valamennyi választógombnak ugyanazzal a *name* értékkel kell rendelkeznie.

Ha valamelyik választási lehetőséget előre felkínáljuk, az adott választógombhoz a *checked* (érték nélkül is használható) jellemzőt kell beírni – természetesen a felhasználó egy másik gombra kattintva a jelölést megváltoztathatja.

Kérdést, előre felkínált választási lehetőséget és autofókuszt is tartalmazó választógombos kódolás és megjelenítése:

```
<form name="gyakorlat">
<p>Ön jelentkezne-e az egyirányú Mars-utazásra?</p>
<p><label><input type="radio" name="valasztogomb" value="igen"
                 checked>igen</label>
   <label><input type="radio" name="valasztogomb" value="nem">
                 nem</label>
   <label><input type="radio" name="valasztogomb" autofocus
                 value="talan">talán</label>
</p>
</form>
```

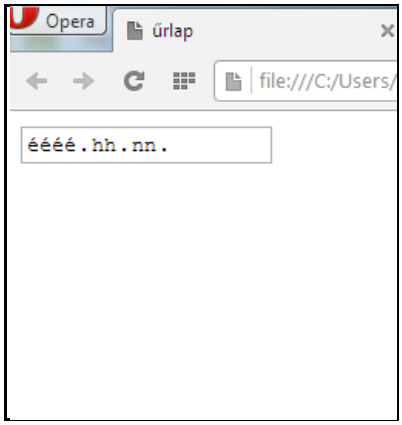


d) Időpont megadására alkalmas űrlapelemek:

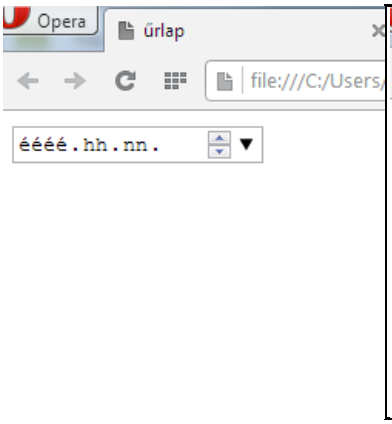
type="date" *type="month"* *type="week"* *type="time"*

A *date*, *month* és *week* értékek a Chrome-ban és Opera-ban ugyanolyan módon adhatók meg:

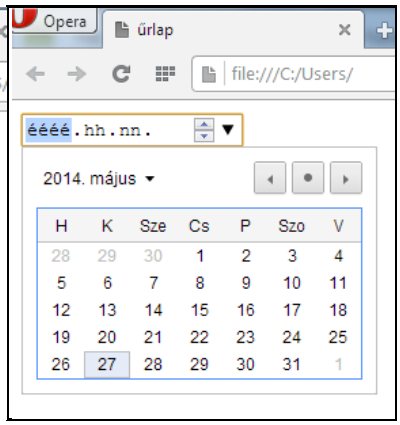
1.) szövegmező jelzi, hogy dátum beállítására szolgál



2.) kurzorral a mezőbe érve beállító nyilak jelennek meg

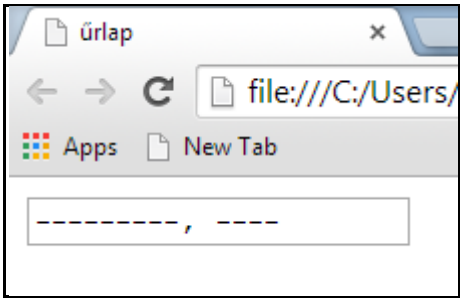


3.) a nagy nyíl naptárat gördít le, a kicsik léptetnek

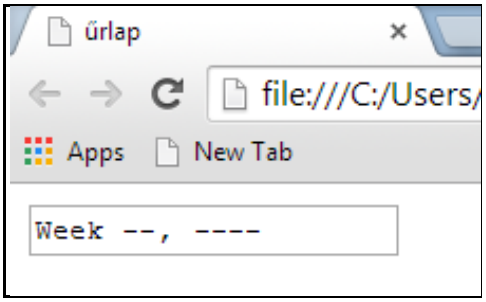


A *month* (hónap) és *week* (hét) esetében a szövegmező kezdő jelzése más, de a továbbiakban megegyezik a *date* (dátum) beállítási módjával:

a *month* kezdő szövegmezője

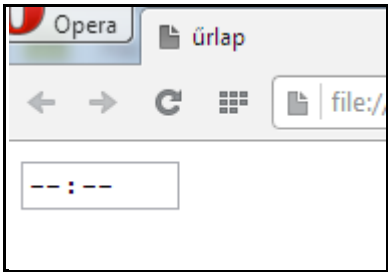


a *week* kezdő szövegmezője

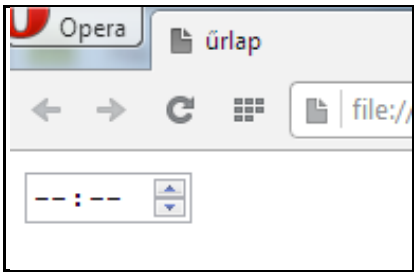


Megjegyzés: Az Internet Explorer, Safari és Firefox az *<input type="date" name= "…."*> kódot kezdő érték nélküli egysoros szövegmezőként (vagyis az *input* alapértelmezett *text* alakjában) jelenítik meg, és kézzel, szövegként kell a mezőbe beírni a dátumot.

A *<input type="time" name="…..">* kódot a Firefox, Safari és Internet Explorer szintén egysoros, kezdőszöveg nélküli szövegmezőként jelenítik meg, definíciószerű űrlapelemet ismét a Chrome és Opera hoz létre. Az idő megadásának lehetőségét jelző szövegmező fölé helyezve a kurzort megjelenik az óra/perc jel és a beállításhoz szükséges két nyíl, melyekre kattintva léptetéssel érhető el a kívánt idő:



a *time* kezdő szövegmezője



a kurzorral a mezőbe érve beállító nyilak jelennek meg

Az időpontok esetében (a gyakorlatban csak az Opera-val) használhatók a *min* és *max* jellemzők:

Ha (például garanciális reklamációra) csak idei dátumot fogadunk el érvényesnek, akkor a kódolás:

```
<input type="date" name="datum" min="2014-01-01">
```

és korábbi időpont bevitelét nem teszi lehetővé az űrlapmező.

Ha (például ügyfélfogadás miatt) csak reggel 8 és délután 4 óra közötti időt tartunk érvényesnek, akkor a kódolás:

```
<input type="time" name="ido" min="08:00" max="16:00">
```

és ezen kívüleső időpont bevitelét nem teszi lehetővé az űrlapmező.

A *step* jellemzővel az alapértelmezett 1 perces és 1 másodperces lépések változtathatók, pl. *step=”600”* esetén 10 perces ugrásokban lehet meghatározni az időt.

e) Mennyiség megadására alkalmas űrlapelemek:

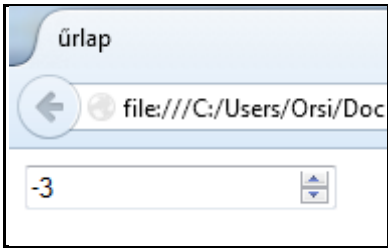
type=”number” *type=”range”*

Egy űrlap szempontjából számnak az tekintendő, ami változtatva is értelmezhető marad – tehát pl. a lakcímben az irányítószám és házszám vagy a bankkártyaszám szövegnek és nem számnak számít, mert növelése vagy csökkentése alapvetően megváltoztatja az értelmét. A számot ezért változtatási lehetőséget hordozó mezőként (*spinbox interface*) kezeli az űrlap, melynek opcionálisan megadható a megengedett legkisebb (*min*) és legnagyobb (*max*) értéke, a lépésenkénti változtatás mértéke (*step*) és a kezdeti számérték (*value*) is, pl.:

```
<input type="number" name="szam" min="-6" max="30" step="3"
       value="-3">
```

Az Internet Explorer a kezdeti értéket mutatja, de utána manuálisan (nem léptetéssel) lehet bevinni a számokat, az Opera és Chrome először csak a kezdő értéket mutatja egy mezőben, majd a kurzort a mező fölé helyezve megjelennek a csökkentést/növelést léptető nyilak.

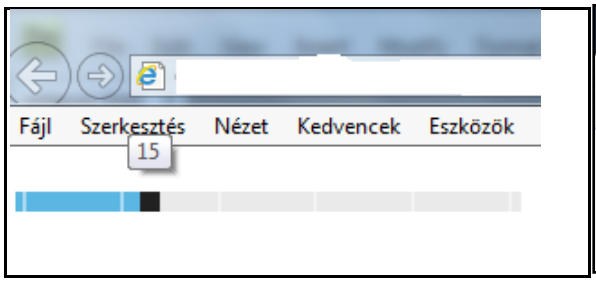
A Firefox viszont egyből a *spinbox*-ot mutatja:

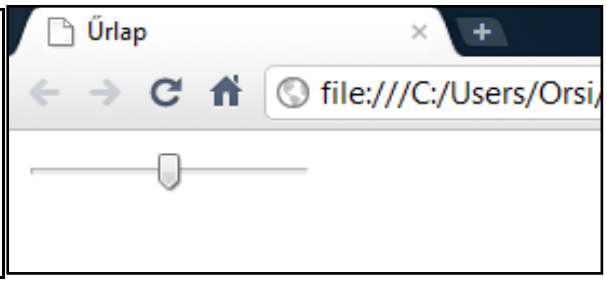


A *range*–el (*range*=tartomány) egy ” –tól” „–ig” értéktartományban való relatív pozíció adható meg. Megjelenítése csúszkával történik, jellemzői megegyeznek a *number*-nél látottakkal. Alapértelmezetten 0 és 100 közötti a tartomány, a kezdőérték a tartományhatárok számtani közepe, a lépésköz pedig 1. Egy kódolási példa egyedi jellemző/értékekkel:

```
<input type="range" name="skala" value="25" min="0" max="50"
       step="5">
```

A legelegánsabban az Internet Explorer jeleníti meg a csúszkát, a többiek nagyon egyszerűek :

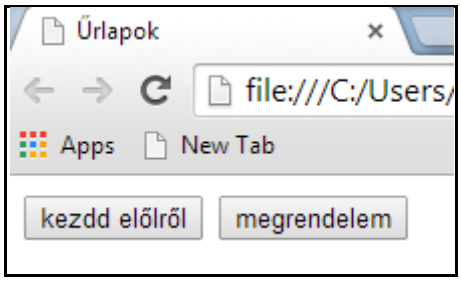




f) „Alaphelyzet” és „Küldés” gombok:

Az űrlap végén helyezkednek el, a *reset*-tel a kezdő állapot állítható vissza, a *submit*-tel pedig a kitöltött űrlap elküldése történik. Alapértelmezett feliratuk a ***value=……*** jellemző/ értékekkel módosítható. A gombok méretét a böngészők a beírt szöveghez igazítják. Például:

```
<input type="reset" name="visszaallitas" value=kezdd elölről>
<input type="submit" name="kuldes" value="megrendelem">
```



g) „Tetszőleges” funkciójú nyomógombot a *<button>……</button>* páros címkével lehet kódolni, *type* jellemzőjének lehetséges értékei *submit, reset,* és *button.*

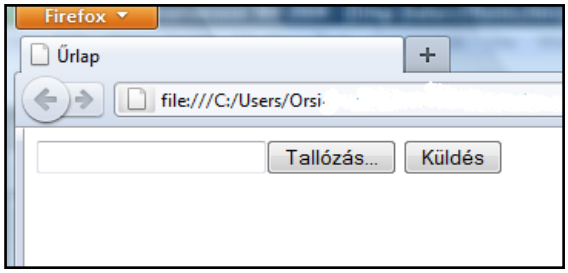
A ***<button type="submit">……</button>*** az elküldést, a *type="reset"*-tel értelemszerűen a kezdő állapot visszaállítását hozza létre. A *type="button"* önmagában nem csinál semmit, a funkciót szkript-nyelvekkel szokás hozzárendelni, ezért a továbbiakban ezzel nem foglalkozunk.

g) Fájlfeltöltés:

Fájlokat (multimédiásakat is) az

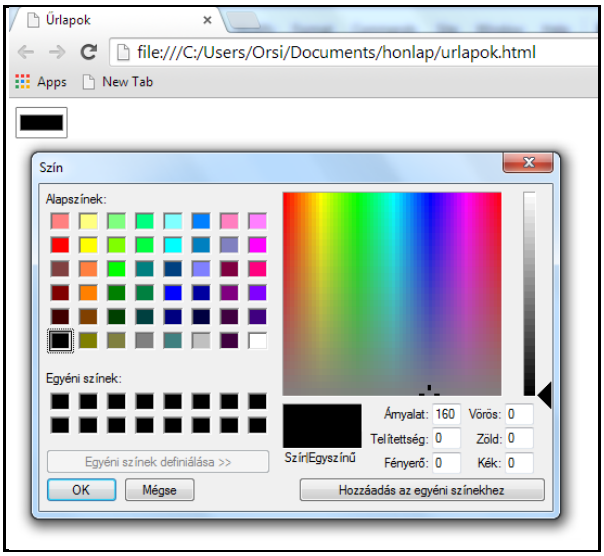
```
<input type="file" name="fajlfeltoltes">
```

kódolással lehet feltölteni, melynek során a felkínált tallózásból választható ki a keresett fájl:



h) Színválasztás:

Az `<input type="color" name="szin">` kódolásra egy fekete téglalapot tartalmazó mező jelenik meg, melyre kattintva a képszerkesztő programoknál megszokott paletta jelenik meg, és abból történhet színválasztást:



Megjegyzés: Az Internet Explorer nem értelmezi a *type="color"* jellemző/értéket.

2.14.2. Egyéb címkék

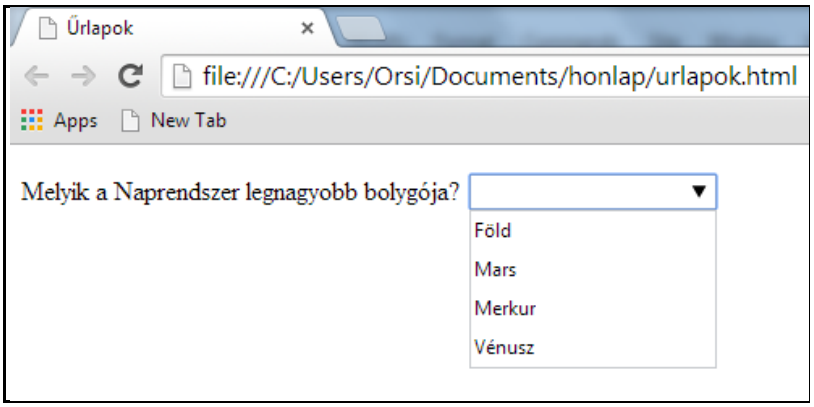
a) A *<datalist>……</datalist>* páros elemmel kódolható egy legördülő lista, melynek elemei *<option value="……..">* sorokkal adhatóak meg. A listát az *<input type="text">* típusú szövegmezőhöz a *list="…."* jellemző/érték pár rendeli hozzá.

A *datalist* funkcióval kódolható a megadott lista egy értékének a kiválasztása, vagy egy - a listában nem szereplő - értéknek a szövegmezőbe való beírása.

Például ha arra a kérdésre kell válaszolni, hogy melyik a Naprendszer legnagyobb bolygója, melyre vagy a néhány felsorolt bolygónévből vagy azokon kívüli megnevezéssel válaszolhat a felhasználó (mint a jelen esetben a helyes válasz ezt kívánja meg), akkor a kódolás:

```
<p><label>Melyik a Naprendszer legnagyobb bolygója?
   <input type="text" name="valasztas" list="bolygok"></label>
</p>
<datalist id="bolygok">
        <option value="Föld">
        <option value="Mars">
        <option value="Merkur">
        <option value="Vénusz">
</datalist>
```

A böngésző a kezdeti állapotban a kérdés után egy normál szövegbeviteli mezőt mutat, melybe az írás céljából a kurzorral belépve legördül az előre felkínált válaszok listája:



A lista kiválasztott elemére kattintva a lista eltűnik és a válasz bekerül a szövegmezőbe. A listától független válasz esetén az első karakter beütésekor szintén eltűnik a lista és a szövegmezőben a beírt szöveg marad. Amennyiben a beírás kezdőbetűje megegyezik valamelyik listelemével, akkor az a listaelem egész addig látható marad, amíg a további betűk bevitelével megszűnik az egyezés.

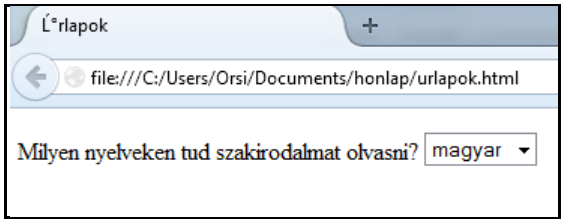
Megjegyzés: A Safari-k nem értelmezik a *datalist* funkciót.

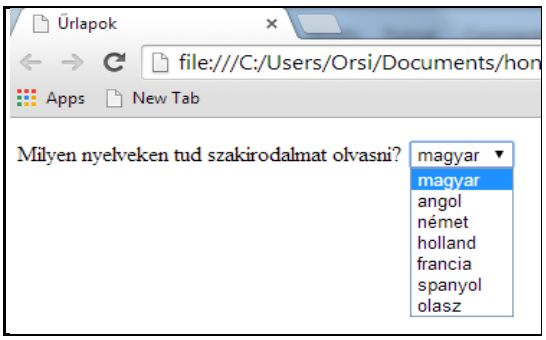
b) A *<select>……</select>* páros címkével kódolható az olyan legördülő lista, amikor csak annak elemeiből lehet egyet vagy többet választani. A lista elemei az *<option>* ……*</option>* páros címkékkel és azok *value* jellemző/érték párjaival adandók meg.

Az idegennyelv ismereti példát még egyszer - ezúttal lenyíló lista formájában - alkalmazva és további nyelvekkel kiegészítve:

```
<p><label>Milyen nyelveken tud szakirodalmat olvasni?
        <select name="nyelv">
                <option value="magyar">magyar</option>
                <option value="angol">angol</option>
                <option value="nemet">német</option>
                <option value="holland">holland</option>
                <option value="francia">francia</option>
                <option value="spanyol">spanyol</option>
                <option value="olasz">olasz</option>
        </select></label>
</p>
```

Alapértelmezetten a felsorolás sorrendjében helyezkednek el a listaelemek, az első a szövegmezőben olvasható, a mellette lévő nyílra kattintva gördül le a többi elemmel a lista, melyből kattintással egy elem választható:

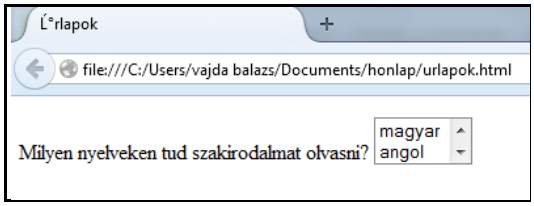


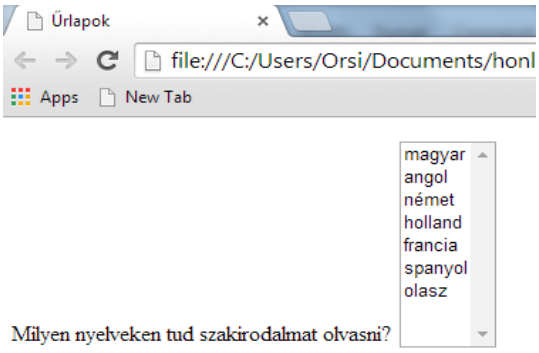


A *select* jellemzőivel a lista jellege alakítható:

- a *size="x"* jellemző/érték párral meghatározható, hogy a listának hány eleme gördüljön le (*x* pozitív egész szám)
- a *selected* jellemzővel (nem szükséges értéket adni) a lista bármelyik eleme beállítható kezdő elemnek
- a *multiple* jellemzővel (nem szükséges értéket adni) több elem is egyidejűleg választható

Példa a *size* jellemző használatára:



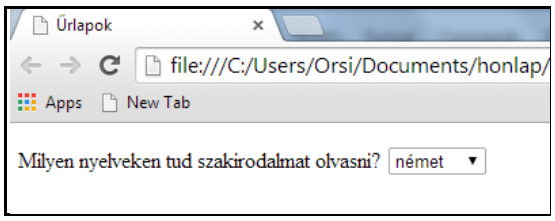


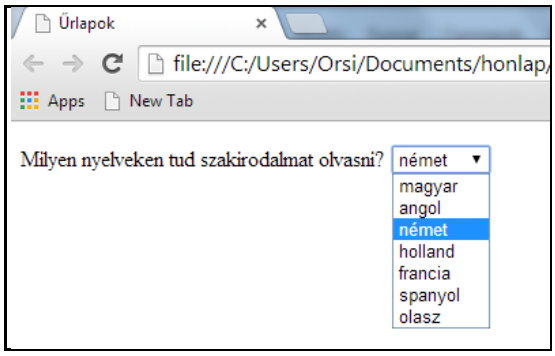
<***select name***="nyelv" ***size***="2"> <***select name***="nyelv" ***size***="9">

A böngészők a *size* jellemző kódolásakor minden esetben görgetősávot alkalmaznak (akkor is, ha annak használatára nincsen szükség). A választási lehetőségeknél nagyobb szám is megadható, ekkor üres sorokat mutatnak a böngészők. A Chrome és Opera a négynél kisebb *size*-méretet nem veszi figyelembe (azaz négyet mindenféleképpen felfed), annál nagyobb értékeket pontosan mutat.

Példa a *selected* jellemző használatára:

```
<p><label>Milyen nyelveken tud szakirodalmat olvasni?
      <select name="nyelv">
            <option value="magyar">magyar</option>
            <option value="angol">angol</option>
            <option value="nemet" selected>német</option>
            <option value="holland">holland</option>
            <option value="francia">francia</option>
            <option value="spanyol">spanyol</option>
            <option value="olasz">olasz</option>
      </select></label>
</p>
```

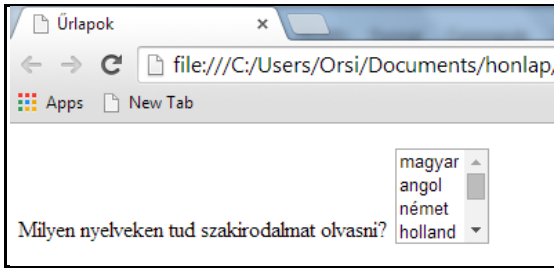




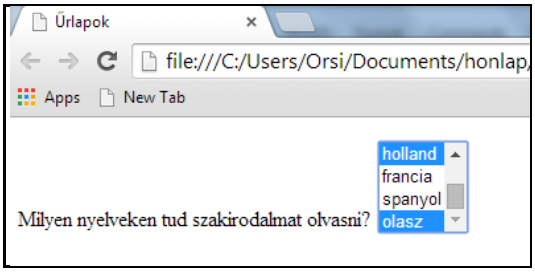
A németet kezdő elemnek kódolva a megjelenítés a németről (nem a magyarról) indul, a nyílra kattintva a német kijelölt marad.

Példa a *multiple* jellemző használatára:

```
<p><label>Milyen nyelveken tud szakirodalmat olvasni?
      <select name="nyelv" multiple>
            <option value="magyar">magyar</option>
            <option value="angol">angol</option>
            <option value="nemet">német</option>
            <option value="holland">holland</option>
            <option value="francia">francia</option>
            <option value="spanyol">spanyol</option>
            <option value="olasz">olasz</option>
      </select></label>
</p>
```



választás előtt



választás után

Több elem egyidejű választhatóságát is lehetővé téve a kinyitott listából lenyomott CTRL billentyű melletti többszöri kattintással több lehetőség választható. A többes választási lehetőséget a görgetősáv jelzi akkor is, amikor az elemek száma ezt egyébként nem indokolja. Többes választás esetén több kezdőérték is megadható.

c) A legördülő lista *<option>* elemei csoportokba is foglalhatók, erre szolgál az *<optgroup>* páratlan címke. Kötelező jellemzője a *label* elemfelirat, hiszen az elemcsoportokat ezek az alcímek azonosítják. A *<select>……<\/select>* lenyíló listában a valamilyen szempont alapján csoportosított *<option>* elemeket tehát *<optgroup label="…">* egyedi nevekkel rendelkező címkék tagolják.

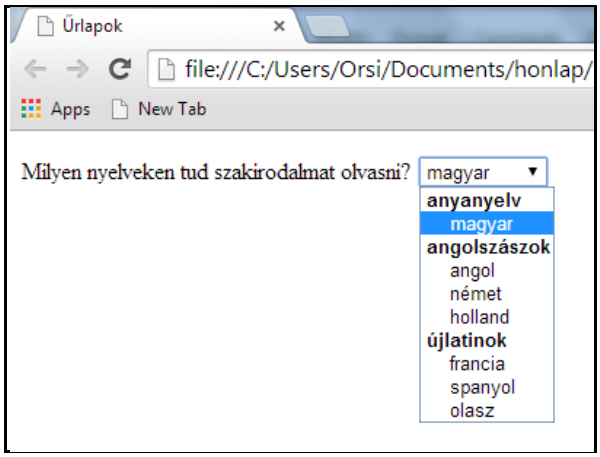
Például a nyelveket taglaló példánknál maradva az alábbi csoportokra osztjuk őket:

anyanyelv – magyar
angolszász nyelvek: angol, német, holland
újlatin nyelvek: francia, spanyol, olasz

A nyelvi csoportokat definiáló kódokat beírva a már sokszor használt kódsorba:

```
<p><label>Milyen nyelveken tud szakirodalmat olvasni?
      <select name="nyelv">
         <optgroup label=anyanyelv>
            <option value="magyar">magyar</option>
         <optgroup label=angolszászok>
            <option value="angol">angol</option>
            <option value="nemet">német</option>
            <option value="holland">holland</option>
         <optgroup label=újlatinok>
            <option value="francia">francia</option>
            <option value="spanyol">spanyol</option>
            <option value="olasz">olasz</option>
      </select></label>
</p>
```

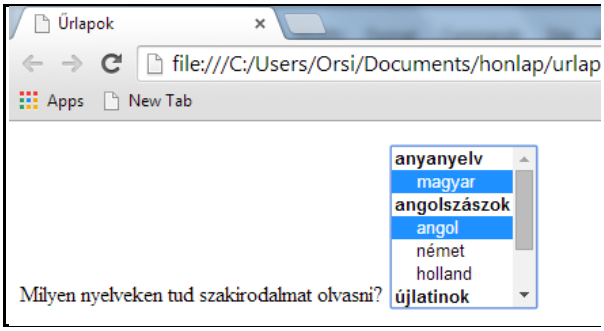
Az alapértelmezett megjelenítés:



Több választást (az alcímek nem jelölhetők ki) a *multiple* jellemzővel, előre kiválasztást a *selected* jellemzővel, egyidejűleg látható listaelemszámot a *size* jellemző/értékkel (ebbe az alcímek is beleszámítanak) definiálhatunk. Például a

```
<select name="nyelv" multiple size="7">
```

kóddal, majd a magyarra és angolra kattintással a megjelenítés:



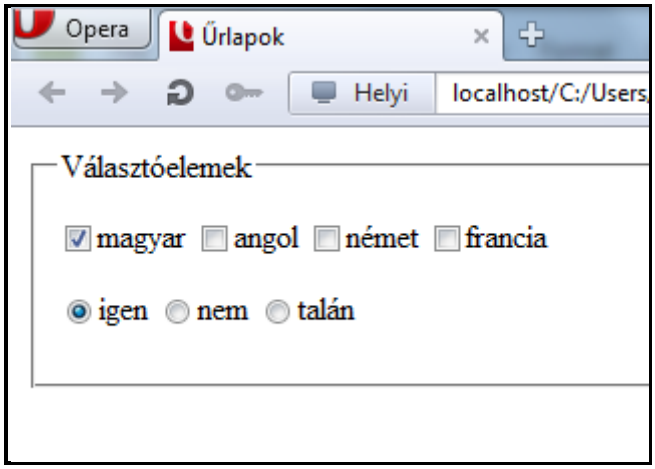
d) Az űrlap elemei a <*fieldset*>…….</*fieldset*> páros címkével (*fieldset*=mezőcsoport) csoportokba foglalhatók, melyeknek <*legend*>……..</*legend*> páros címkével (*legend* = felirat, elemcsoportnév, mezőcsoportnév, jelmagyarázat) név is adható.

Pl. *Választóelemek* címmel egy elemcsoportba foglalva a korábban használt jelölőnégyzeteket és választógombokat, az alábbi kódolás adódik:

```
<fieldset>
<legend>Választóelemek</legend>
   <p>Milyen nyelveken tud szakirodalmat olvasni?</p>
   <p><label><input type="checkbox" name="nyelv1"
                    value="magyar" checked>magyar</label>
      <label><input type="checkbox" name="nyelv2"
                    value="angol">angol</label>
      <label><input type="checkbox" name="nyelv3"
                    value="nemet">német</label>
      <label><input type="checkbox" name="nyelv4"
                    value="francia">francia</label>
   </p>
   <p>Ön jelentkezne-e az egyirányú Mars-utazásra?</p>
```

```
    <p><label><input type="radio" name="valasztogomb"
                     value="igen" checked>igen</label>
       <label><input type="radio" name="valasztogomb"
                     value="nem">nem</label>
       <label><input type="radio" name="valasztogomb"
                     value="talan">talán</label>
    </p>
</fieldset>
```

A *fieldset* kerettel a teljes vízszintes teret kitöltik alapértelmezetten a böngészők, ezért az elemcsoport szélességét majd CSS-ben lehet adott méretre beállítani:

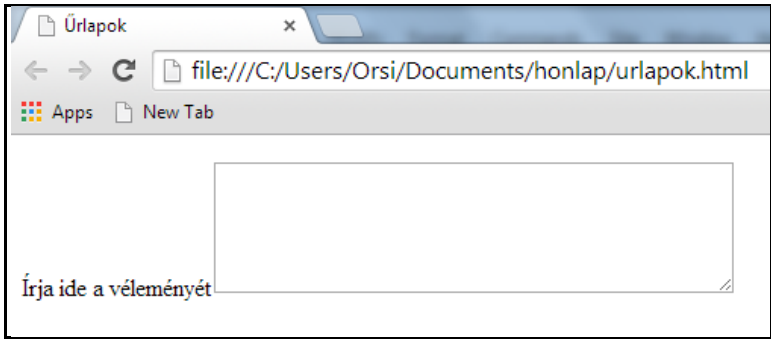


e) Hosszabb, többsoros (sortörést is megengedő) szövegmezőt hoz létre a *<textarea>* ……..*</textarea>* páros címke. Elemfelirata kialakítására ismét a *label* páros címke használható, alapértelmezett mérete: kb. 20 karakter hosszú és 2 sor magas. A szövegmező méretét a *cols* (cols=*columns*=oszlopok) és *rows* (*rows*=sorok) jellemzőkkel magunk is megadhatjuk, ezen felül a Chrome-nál, Firefox-nál és Opera-nál a szövegmező sarka a kurzorral nagy méretre kihúzható. Ha egy sor megtelik a beírt szöveggel, automatikus sortöréssel folytatódik. A sorok számát meghaladva görgősáv jelenik meg a szövegmező jobb szélén, és a szöveg beírása folytatható (az Internet Explorer-nél kezdettől fogva van görgetősáv).

Például egy 40 karakter hosszú, 5 karakter magasságú szövegmező kódolása *Írja ide a véleményét* felirattal:

```
<p><label>Írja ide a véleményét<textarea name="velemeny"
          cols="40" rows="5"></textarea></label></p>
```

A megjelenítés:



A hosszú szövegmezőbe is kódolható kezdőszöveg a *placeholder* jellemző/értékkel, és korlátozható a felhasználó által bevihető karakterek száma a *maxlength* jellemző/értékkel. A *maxlength* értéke független a szövegdoboz méretétől, lehet rövidebb és hosszabb is. Például:

```
<p><label>Írja ide a véleményét<textarea name="velemeny"
     cols="40" rows="5" placeholder="De röviden, tömören, max.
     150 karakterrel!" maxlength="150"></textarea></label></p>
```

Megjegyzések az Űrlapok fejezethez:

- valamennyi űrlapkódnak a *<form name="……">* címkével kell kezdődnie, és a *</form>* címkével zárulnia – ezt az esetek nagy többségében a kódolási példákban nem írtuk ki
- a teljes HTML dokumentumhoz szükséges *<!doctype html><html lang="hu">……* *</html>* kezdést és befejezést ugyancsak nem tüntettük fel
- a kötelezően kitöltendő űrlapelemeket a *required* jellemzővel (nem kell értéket adni) lehet megjelölni – ennek az űrlap elküldésekor van szerepe, ami kívül esik ennek a jegyzetnek a tárgykörén
- a következő példa űrlapban a *<style>……</style>* elemben egy CSS kódot is alkalmaztunk, ennek értelmezése a CSS fejezetben történik meg

2.14.3. Egy űrlap megszerkesztése

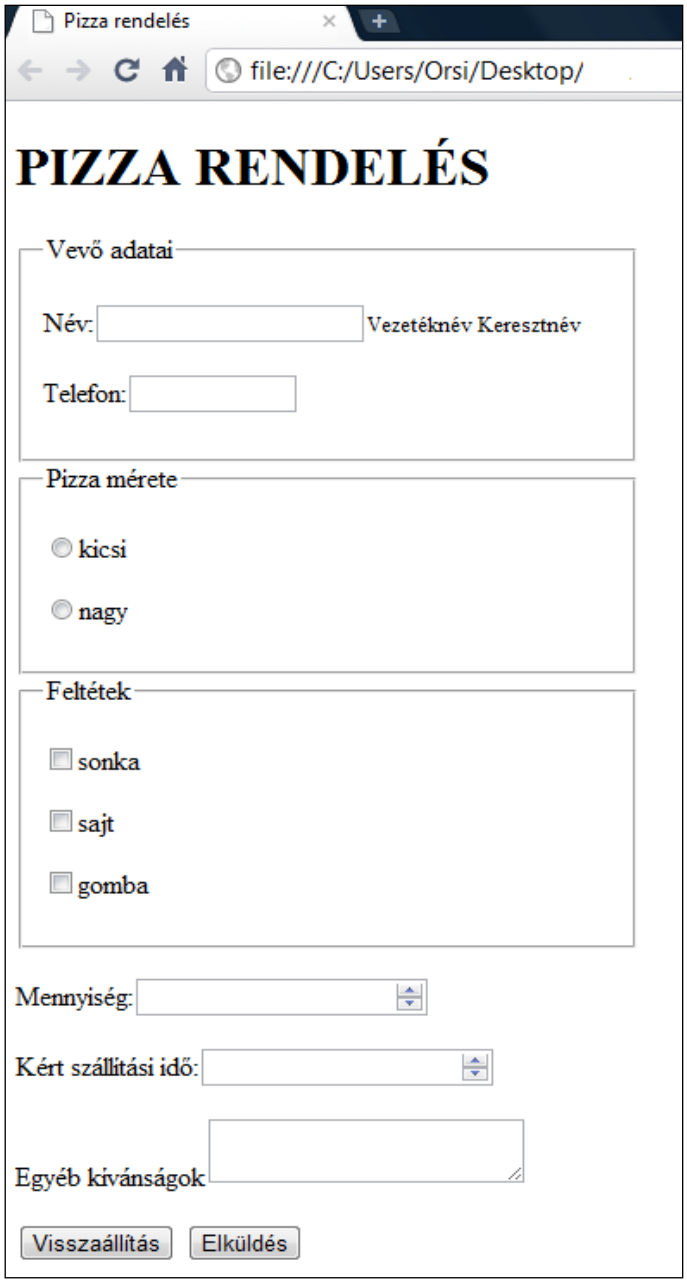
Az űrlapelemek egy alkalmazása pl. egy képzelt pizza-rendelési űrlapon bemutatva:

```
<!doctype html>
<html lang="hu">
   <head>
      <meta charset="utf-8">
      <title>Pizza rendelés</title>
      <style>
         fieldset { width:330px; }
      </style>
   </head>
   <body>
      <form name="rendeles">
         <h1>PIZZA RENDELÉS</h1>
         <fieldset>
         <legend>Vevő adatai</legend>
         <p><label>Név:<input type="text" name="nev">
            <small>Vezetéknév Keresztnév</small></label></p>
         <p><label>Telefon:<input type="tel" name="telefon"
             size="10"></label></p>
         </fieldset>
```

```
        <fieldset>
        <legend>Pizza mérete</legend>
        <p><label><input type="radio" name="meret"
           value="kicsi">kicsi</label></p>
        <p><label><input type="radio" name="meret"
           value="nagy">nagy</label></p>
        </fieldset>
        <fieldset>
        <legend>Feltétek</legend>
        <p><label><input type="checkbox" name="feltet1"
           value="sonka">sonka</label></p>
        <p><label><input type="checkbox" name="feltet2"
           value="sajt">sajt</label></p>
        <p><label><input type="checkbox" name="feltet3"
           value="gomba">gomba</label></p>
        </fieldset>
        <p><label>Mennyiség:<input type="number"
           name="mennyiseg"></label></p>
        <p><label>Kért szállítási idő:<input type="time"
           name="kiszallitas"></label></p>
        <p><label>Egyéb kívánságok<textarea name="egyeb">
           </textarea></label></p>
        <input type="reset" name="visszaallitas">
        <input type="submit" name="kuldes">
      </form>
    </body>
</html>
```

A színkódok a szerkezeti beazonosítást hivatottak segíteni. A fej részben a <*style*> ...</*style*> elemben a *fieldset* szélességet CSS-ben adtuk meg, valamennyi egyéb tulajdonság alapértelmezetten jelenik meg.

Megjegyzés: Az űrlap szerkezete és az egyes elemek fő kialakítása kötött, de a betűk, szegélyek, margók, színek, háttérszínek és háttérképek, szélességek és az elemek relatív helyzete a CSS részben leírtak szerint formázhatók.



Az űrlapoknál felhasznált HTML címkék:

Cím:	Funkciója:	
<form>	űrlapot definiál	páros címke
<input>	információbeviteli vagy választást lehetővé tevő mezőt definiál	páratlan címke
<label>	elemfeliratot definiál	páros címke
<fieldset>	űrlap elemek csoportjait definiálja	páros címke
<legend>	űrlap elemcsoport címet definiál	páros címke
<select>	legördülő listát definiál	páros címke
<option>	legördülő listaelemet definiál	páros címke
<textarea>	többsoros szövegmezőt definiál	páros címke
<button>	tetszőleges nyomógombot definiál	páros címke
<datalist>	szövegmezőhöz tartozó legördülő listát definiál	páros címke
<optgroup>	listaelem csoportokat definiál	páratlan címke

2.15. Meta-adatok

Visszatérve *„A weboldal szerkezete”* fejezetben megkezdett bevezetésre, ill. kibővítve az ott leírtakat, a főbb tudnivalók az alábbiakban foglalhatók össze:

A meta-adatok a dokumentummal kapcsolatos különféle jellemzők és utasítások tárolására szolgálnak. A weboldal fej részében helyezkednek el, tehát a böngészők nem jelenítik meg őket, kivéve a címet, amely a böngészők címsorában (ill. a keresők találatainak felsorolásában és a Kedvencek/Könyvjelzők menüben) jelenik meg. Néhány meta-adatnak saját címkéje van (*title*=cím, *link*=kapocs, *style*=stílus) a többi a *meta* címke jellemzőjeként adható meg.

Kötelező meta-adatok a *meta charset=”utf-8”* (vagy *iso-8859-2)* karakterkészlet, a *title* (weblap címe), a *link* (a stíluslap csatolása - ha van külső stíluslap) és a *style* (ha van belső stílus definíció).

Egy stílust mindenféleképpen célszerű megadni – bár a HTML oldalt a böngészők akkor is minden információval, logikus struktúrában alapértelmezett tulajdonságaikkal megjelenítik, ha nincsen hozzá stílus definiálva, de így a weblap egy sematikus, színtelen váz lesz.

Nem kötelező, de minden esetben javasolt meta-adatok:

<meta name=”keywords” content=”……”> páratlan címke

meta a címke, *name* (név) és *content* (tartalom) a jellemzők, *keywords* (kulcsszavak/keresőszavak) és a ténylegesen megadott kulcsszavak a jellemzők értékei

A keresőszavak/kulcsszavak megadása a keresők helyes találatát segíti elő. Sok, a tartalomra jellemző szó megadása célszerű, az ékezetes szavakat felsorolásszerűen, vesszővel elválasztva lehet megadni.

<meta name=”description” content=”……> páratlan címke

meta a címke, *name* (név) és *content* (tartalom) a jellemzők, *description* (leírás) és a leírás szövege a jellemzők értékei

A leírás a weboldal rövid tartalmi összefoglalása, a keresők találati listáján az oldal rövid leírásaként jelenik meg.

A jelen esetben alkalmazható kulcsszavak pl. „HTML5, CSS3, webszerkesztés”, rövid tartalom pl. „Szabványkövető weblap készítése”. Ennek kódolása:

```
<meta name=”keywords” content=”HTML5, CSS3, webszerkesztés”>
<meta name=”description” content=”Szabványkövető weblap
      készítése”>
```

Egyéb meta-adatok:

A keresőprogramok (kereső robotok) indexelik a webhelyek teljes tartalmát, azaz elolvassák őket és tárgymutatót készítenek róluk. Amikor a felhasználók rákeresnek egy szóra, kifejezésre, címre, idézetre, stb., a keresők átnézik az indexelt adatbázisukat, és azokat a webhelyeket adják meg találatként, melyekben szerepel a keresett adat.

a) Ha van olyan tartalom egy webhelyen, amit a tulajdonosa nem akar indexeltetni (pl. még nincs teljesen kész, nem mindenki számára publikus családi/magánjellegű információt tartalmaz, stb.), bizonyos fájlok kizárhatók az indexelésből.

A kereső robotok a webhely címe után először azt ellenőrzik, van-e robotkizáró fájl megadva. Ha igen, analizálják a tartalmát – indexelhetik-e a webhely tartalmát vagy elemezhetik-e *link*-ek céljából, vagy sem.

A robotkizáró fájl egy *robots.txt* szövegfájl, ebben felsorolhatók azok a mappák és keresők, melyekre a megkötések vonatkoznak. Pl.:

<meta name="robots" content="`noindex, nofollow`***">***
sem nem indexelhetők, sem link-ek szempontjából nem analizálhatók

<meta name="robots" content="`index, all`***">***
indexelhetők és analizálhatók

user-agent:* (helyettesítő karakter)
minden keresőt távol tart

disallow: / fájl neve
az adott fájlt letiltja

b) A *meta* címke jellemzőivel a weblapra vonatkozó számos további információ adható meg, pl. ki készítette a weblapot (*author*), milyen szoftverrel készült (*generator*), a készítő program azonosítója (*progid*), mikor volt utoljára frissítve (*revised*), hány naponként indexelje (*revisited-after*), szerzői jogok (*copyright*), a terjesztés köre (teljes web/csak intranet) (*distribution*), a weboldal lejárati ideje (*expires*), a webhely tulajdonosa (*owner*), másodlagos tartalmi leírás (*abstract*), stb. Pl.:

```
<meta name="author" content="nagy programozó">
<meta name="distribution" content="web">
<meta name="copyright" content="2012©nagyprogramozó">
```
stb.

2.16. Webhely hierarchikus mappaszerkezettel

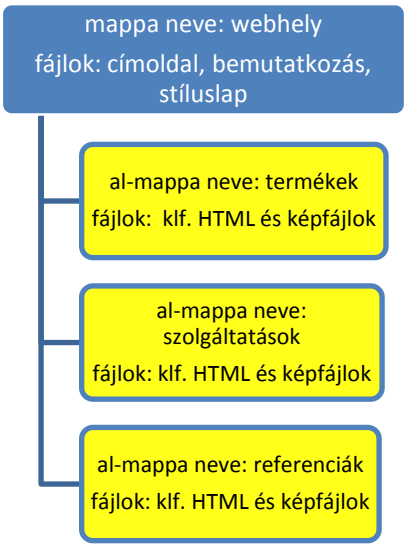
Amikor az első („Alapok”) fejezetben egy webhely kialakítása legegyszerűbb esetének ismertetését elkezdtük, a következőt írtuk:

„A számítógépben egy külön (al-mappákat nem tartalmazó) mappában célszerű a weboldalak valamennyi alkotóelemének (a tartalmat hordozó HTML oldalaknak, a formázást adó CSS stíluslap(ok)nak, a felhasználandó multimédiás elemeknek) a gyűjtése.”

A fenti kiindulási mód leegyszerűsíti a weblapszerkesztés tanulását és egyszerűbb webhelyek kialakításakor tökéletesen megfelelő. Nagy méretű, bonyolult szerkezetű webhelyek esetén viszont növeli az áttekinthetőséget, ha a nagyszámú fájlokat valamilyen rendezőelv szerint al-mappákban csoportosítjuk.

Multimédiában gazdag webhely esetében a csoportosítás történhet pl. a hang- álló- és mozgókép fájlok kategóriái alapján:

Egy cég kiterjedt tevékenységét ismertető webhely mappaszerkezete ugyanakkor pl. tematikus kategóriák alapján alakítható ki:

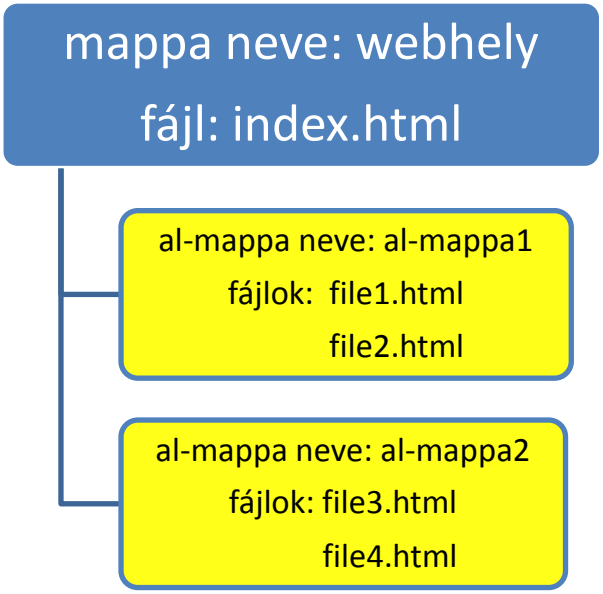


Az egyes al-mappák mindegyik esetben szükség esetén további al-al-mappákra is tagolhatók. Nem helyezhető viszont semmilyen al-mappába a webhely kezdőlapja, melynek *index.html* a fájlneve és kiterjesztése.

Az egységes gyűjtőmappa helyetti tagolt mappaszerkezet a webhely egyes fájljainak egymásra vonatkozó hivatkozási módját befolyásolja. Az elérési (hivatkozási) útvonalak mentén al-mappákból kellhet kilépni, a mappa-szerkezetben feljebb vagy lejjebb haladni és esetleg egy másik al-mappába újra belépni.

Windows operációs rendszerben a **..** (pont-pont) az al-mappából kilépést, ill. egy szinttel feljebb lépést, a / (törtvonal) a belépést egy másik al-mappába, ill. az al-mappanév és fájlnév egymástól való elválasztását jelenti.

A hivatkozások kialakításának bemutatására az alábbi mappa-szerkezetet használjuk:



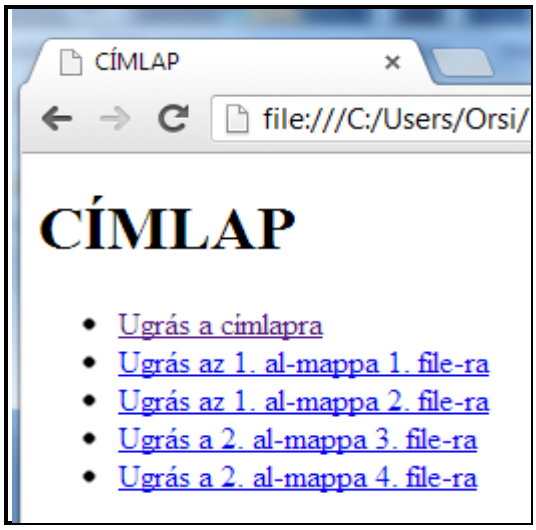
A kezdőlap nem kerülhet al-mappába, stíluslap útján a formázással/elrendezéssel most nem foglalkozunk. Mind a kezdőlapra, mind a két al-mappában lévő két-két fáljba ugyanazt a navigációs listát helyezzük, és az elérési útvonalak bemutatása céljából minden fáljból minden másik fájlba való közvetlen hivatkozást (ugrási lehetőséget) hozunk létre. Az egyes fájlok bármilyen tartalommal rendelkezhetnek, ezért most tartalommal nem foglalkozunk.

A cím és a navigációs lista HTML kódjának váza minden weboldalon egységes (az egyes fájlok önmagukra hivatkozásait is célszerű meghagyni az áttekinthetőség érdekében):

```
<!doctype html>
<html lang="hu">
   <head>
      <meta charset="utf-8">
      <title>…............</title>
      <style>…............</style>
   </head>
   <body>
      <h1>……................</h1>
      <ul>
         <li><a href="…........html">Ugrás a címlapra</a></li>
         <li><a href="…........html">Ugrás az 1. al-mappa 1. file-
             ra</a></li>
         <li><a href="…........html">Ugrás az 1. al-mappa 2. file-
             ra</a></li>
         <li><a href="…........html">Ugrás a 2. al-mappa 3. file-
             ra</a></li>
         <li><a href="…........html">Ugrás a 2. al-mappa 4. file-
             ra</a></li>
      </ul>
   </body>
</html>
```

Az egyes oldalakon – hogy világosan lássuk, hol vagyunk éppen - a *title*-be és *h1*-be a CÍMLAP, 1. AL-MAPPA 1. FILE, 1. AL-MAPPA 2. FILE, 2. AL-MAPPA 3. FILE, ill. 2. AL-MAPPA 4. FILE cím kerül.

Fentiek alapján pl. a címlapot így mutatják a böngészők:



A hivatkozásokra kattintva még hibaüzenetet kapunk, hiszen egyetlen érvényes ugrási célpontot sem kódoltunk.

A fájloknak önmagukra való hivatkozását, ill. a velük egy al-mappában lévő fájlokra hivatkozást a korábban is alkalmazott módon, *fájlnév.kiterjesztés*–ként lehet megadni, hiszen nem kell al-mappából ki- belépni, a mappaszerkezetben szintet változtatni (a kódolásban kékkel jelölve).

a) A címlap az egyes al-mappákkal közös mappában van, tehát az *index*-ből indulva nem kell mappából kilépni vagy szintet változtatni – csak be kell lépni az egyes al-mappákba, aztán az egyes fájlokba (a kódolásban zölddel jelölve):

```
<!doctype html>
<html lang="hu">
   <head>
      <meta charset="utf-8">
      <title>CÍMLAP</title>
   </head>
   <body>
      <h1>CÍMLAP</h1>
      <ul><li><a href="index.html">Ugrás a címlapra</a></li>
          <li><a href="al-mappa1/file1.html">Ugrás az 1. al-
                mappa 1. file-ra</a></li>
          <li><a href="al-mappa1/file2.html">Ugrás az 1. al-
                mappa 2. file-ra</a></li>
          <li><a href="al-mappa2/file3.html">Ugrás a 2. al-
                mappa 3. file-ra</a></li>
          <li><a href="al-mappa2/file4.html">Ugrás a 2. al-
                mappa 4. file-ra</a></li>
      </ul>
   </body>
</html>
```

b) Az al-mappákban lévő fájlokból a címoldalra vagy a másik al-mappában lévő fájlokba az adott al-mappából kilépve és az *index*-be vagy a másik al-mappába majd fájlba belépve jutunk el (a kódolásban pirossal jelölve):

```
<!doctype html>
<html lang="hu">
   <head>
```

```
        <meta charset="utf-8">
        <title>1. AL-MAPPA 1. FILE</title>
    </head>
    <body>
        <h1>1. AL-MAPPA 1. FILE</h1>
        <ul>
            <li><a href="../index.html">Ugrás a címlapra</a></li>
            <li><a href="file1.html">Ugrás az 1. al-mappa 1.
                    file-ra</a></li>
            <li><a href="file2.html">Ugrás az 1. al-mappa 2.
                    file-ra</a></li>
            <li><a href="../al-mappa2/file3.html">Ugrás a 2. al-
                    mappa 3. file-ra</a></li>
            <li><a href="../al-mappa2/file4.html">Ugrás a 2. al-
                    mappa 4. file-ra</a></li>
        </ul>
    </body>
</html>
```

Az 1. al-mappa 2. fájl hivatkozásai az 1. al-mappa 1. fájl-éval megegyeznek:

```
<!doctype html>
<html lang="hu">
    <head>
        <meta charset="utf-8">
        <title>1. AL-MAPPA 2. FILE</title>
    </head>
    <body>
        <h1>1. AL-MAPPA 2. FILE</h1>
        <ul>
            <li><a href="../index.html">Ugrás a címlapra</a></li>
            <li><a href="file1.html">Ugrás az 1. al-mappa 1.
                    file-ra</a></li>
            <li><a href="file2.html">Ugrás az 1. al-mappa 2.
                    file-ra</a></li>
            <li><a href="../al-mappa2/file3.html">Ugrás a 2. al-
                    mappa 3. file-ra</a></li>
            <li><a href="../al-mappa2/file4.html">Ugrás a 2. al-
                    mappa 4. file-ra</a></li>
        </ul>
    </body>
</html>
```

A 2. al-mappa 3. fájl hivatkozásai a fentiek analógiájára:

```
<!doctype html>
<html lang="hu">
    <head>
        <meta charset="utf-8">
        <title>2. AL-MAPPA 3. FILE</title>
    </head>
    <body>
        <h1>2. AL-MAPPA 3. FILE</h1>
        <ul>
            <li><a href="../index.html">Ugrás a címlapra</a></li>
            <li><a href="../al-mappa1/file1.html">Ugrás az 1. al-
                        mappa 1. file-ra</a></li>
            <li><a href="../al-mappa1/file2.html">Ugrás az 1. al-
                        mappa 2. file-ra</a></li>
```

```
            <li><a href="file3.html">Ugrás a 2. al-mappa 3. file-
                        ra</a></li>
            <li><a href="file4.html">Ugrás a 2. al-mappa 4. file-
                        ra</a></li>
        </ul>
    </body>
</html>
```

A 2. al-mappa 4. fájl hivatkozásai a 2. al-mappa 3. fájléval azonosak (a kódolásban csak a címet és h1-et kell 4. FILE-ra átírni).

Fenti elérési utak beírásával minden fájlból minden fájlba tudunk ugrani a hierarchikus mappaszerkezetünkben.

Fontos: Az egyes fájlokból az azonos webhelyen belüli másik fájlra mutató elérési út fenti (viszonyított) megadását *relatív hivatkozásnak* nevezik. Amennyiben a számítógépen belül áthelyezzük a webhely mappáját, vagy szerver-re tesszük fel, vagy egyik szerver-ről egy másikra helyezik át, a webhely fájljainak egymáshoz viszonyított helyzete változatlan, a hivatkozások érvényesek maradnak.

A külső webhelyekre/weboldalakra/fájlokra vonatkozó hivatkozásokat a teljes elérési úttal, azaz *abszolút hivatkozással* (http://www.....) kell megadni.

Megjegyzés: a) amennyiben homogén gyűjtőmappában tároljuk a különböző fájlokat, elmarad a ki- belépés és szintváltozás (lásd az azonos al-mappában lévő fájlok esetét), és visszakapjuk az egyszerű *fájlnév.kiterjesztés* elérési utakat.

b) Az <***a href=”****fájlnév****.html”***> *hivatkozott weboldal neve, szó, fogalom, és/vagy kép* <***/a***> hivatkozáson kívül al-mappák esetén az elérési út kialakítását érinti:

állóképeknél: ***<img src=”…….fájlnév.kiterjesztés”>***

mozgóképeknél: ***<video src="……..fájlnév.kiterjesztés"></video>*** vagy

```
<video>
        <source src="……fájlnév.kiterjesztés1">
        <source src="……fájlnév.kiterjesztés2">
        <source src="……fájlnév.kiterjesztés3">
</video>
```

hangnál: ***<audio src="…….fájlnév.kiterjesztés"></audio>*** vagy

```
<audio>
        <source src="……fájlnév.kiterjesztés1">
        <source src="……fájlnév.kiterjesztés2">
</audio>
```

képtérképnél, amennyiben webhelyen belüli az ugrási célpont

```
<map id="……">
  <area shape="……" coords="……" title="……" alt="……" href="…….">
  <area shape="……" coords="……" title="……" alt="……" href="…….">
  <area shape="……" coords="……" title="……" alt="……" href="…….">
  <area shape="……" coords="……" title="……" alt="……" href="…….">
</map>

<img src="……képfájl neve.kiterjesztése"usemap="#map
          címkében megadott id">
```

2.17. HTML5 összefoglaló

2.17.1. A statikus weblapok strukturált tartalmának elemi építőkockái a címsorok, bekezdések, listák, táblázatok és beágyazott multimédia elemek (álló- és mozgókép, hang, webtartalom), melyeket a dokumentumra vonatkozó meta-adatok, a tartalom értelmezését és formázását segítő elemek, és a dokumentumban, a webhelyen vagy a világhálós hipertérben való mozgást lehetővé tevő hivatkozások egészítenek ki. Az űrlapok kialakítása ugyancsak HTML-el történik, de működtetésükhöz interaktivitást lehetővé tevő szkript-nyelvek használata szükséges.

A statikus weblapokban használható (régi és új) HTML5 címkék több szempont szerint csoportosíthatók:

a) **Alapértelmezett megjelenítés szerint** a címkék által létrehozott elemek lehetnek sorban elrendezettek (*inline*) vagy blokkszintűek (*block*). A megjelenítés és elrendezés megváltoztatása CSS-el történik.

b) **Formai szempontból** a címkék lehetnek párosak vagy páratlanok. Nagy többségük páros, páratlan címkék az: *area, base, br, col, embed, hr, img, input, link, meta, optgroup, param, source, wbr.*

c) **Funkciójuk alapján** az alábbiak szerint csoportosíthatók:

0. dokumentumtípus deklaráció és megjegyzés: *<!doctype>* és *<!--……--->*
1. gyökérelem: *html*
2. metaadatok megadása: *base, head, link, meta, style, title*
3. szakaszolás: *address, article, aside, body, footer, h1-h6, header, main, nav, section*
4. tartalom csoportosítása: *blockquote, br, p, hr, dd, div, dl, dt, figcaption, figure, li, ol, pre, ul*
5. szöveg jelentése: *a, abbr, b, cite, code, dfn, em, i, kbd, mark, meter, q, s, samp, small, span, strong, sub, sup, time, u, var, wbr*
6. szöveg szerkesztése: *del, ins*
7. táblázatok kialakítása: *caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr*
8. beágyazott tartalom: *area, audio, embed, iframe, img, map, object, param, source, track, video*
9.űrlap: *button, datalist, fieldset, form, input, label, legend, optgroup, option, select, textarea*
10. a távol-keleti nem latin betűs nyelvek karaktereihez: *ruby, rt, rp*
11. a közel-keleti nem balról jobbra de vízszintesen írt nyelvekhez: *bdi, bdo*

d) **Jelentésük szerinti** összefoglalás abc-szerinti sorrendben:

címke		***fejezet***	***jelentése***
<!-- -->	páratlan	2.4	megjegyzés beszúrása, kódrészlet ideiglenes kiiktatása
<!doctype>	páratlan	2.2	dokumentumtípus definíció
<a>	páros	2.13	horgony, hivatkozást és ugrási célpontot definiál
<abbr>	páros	2.11	rövidítést, mozaikszót definiál
<address>	páros	2.11	dokumentum szerzőjének elérhetőségét definiálja
<area>	páratlan	2.13	hivatkozások részére területet definiál
<article>	páros	2.12	egy *section*-on belüli kisebb tartalmi elemet definiál
<aside>	páros	2.12	gondolatmenethez érintőlegesen csatlakozó részt definiál
<audio>	páros	2.8	hangot definiál
<b>	páros	2.11	félkövér szöveget definiál

<blockquote>	páros	2.11	idézetblokkot definiál
<body>	páros	2.2	a dokumentum törzsét definiálja
 	páratlan	2.4	sortörést hoz létre
<button>	páros	2.14	tetszőleges nyomógombot definiál
<caption>	páros	2.6	táblázat címet definiál
<cite>	páros	2.11	címhivatkozást definiál
<col>	páratlan	2.6	táblázat oszlopo(ka)t definiál
<colgroup>	páros	2.6	táblázat oszlopcsoportot definiál
<datalist>	páros	2.14	szövegmezőhöz tartozó legördülő listát definiál
<dd>	páros	2.5	meghatározást/társítást definiálja
<del>	páros	2.11	törölt szöveget definiál
<dfn>	páros	2.11	meghatározást definiál
	páros	2.12	blokkszintű tároló (befoglaló) elemet definiál
<dl>	páros	2.5	meghatározás (definíciós) listát definiál
<dt>	páros	2.5	meghatározandó szót/fogalmat/kifejezést definiál
	páros	2.11	kihangsúlyozást definiál
<embed>	páros	2.8	beágyazott multimédia elemet definiál
<fieldset>	páros	2.14	űrlap elemek csoportjait definiálja
<figcaption>	páros	2.12	beágyazott elemek címét definiálja
<figure>	páros	2.12	beágyazott elemek és címük csoportját definiálja
<footer>	páros	2.12	weboldal láblécét definiálja
<form>	páros	2.14	űrlapot definiál
<h1>……<h6>	páros	2.4	címsorokat definiál
<head>	páros	2.2	dokumentumra vonatkozó információkat definiál
<header>	páros	2.12	weboldal fejlécét definiálja
<hr>	páratlan	2.4	vízszintes elválasztó vonalat szúr be
<html>	páros	2.2	HTML dokumentumot definiál
<i>	páros	2.11	dőltbetűs szöveget definiál
<iframe>	páros	2.11	szövegközi keretet (beágyazott tartalmat) definiál
<image>	páratlan	2.7	(álló)képet definiál
<input>	páratlan	2.14	beviteli vagy választást lehetővé tevő mezőt definiál
<ins>	páros	2.11	beszúrt szöveget definiál
<kbd>	páros	2.11	billentyűzet karaktert definiál
<label>	páros	2.14	elemfeliratot definiál
<legend>	páros	2.14	űrlap elemcsoport címet definiál
	páros	2.5	lista elemet definiál
<link>	páratlan	2.2	a dokumentum és egy külső forrás kapcsolatát definiálja
<main>	páros	2.12	a dokumentumtörzs fő tartalmi részét jelöli ki
<map>	páros	2.13	képtérképet definiál
<mark>	páros	2.11	szövegrész kiemelését definiálja
<meta>	páratlan	2.2	a dokumentumra vonatkozó meta-adatokat definiál
<meter>	páros	2.11	egy ismert tartományon belüli értéket definiál
<nav>	páros	2.12	weboldal navigációs részét definiálja
<object>	páros	2.8	beágyazott multimédia elemet definiál
	páros	2.5	számozott (rendezett) listát definiál
<optgroup>	páratlan	2.14	űrlap legördülő listaelem csoportokat definiál
<option>	páros	2.14	legördülő listaelemet definiál
<p>	páros	2.4	bekezdést definiál
<param>	páratlan	2.8	multimédiás objektum beállításait definiálja
<pre>	páros	2.11	előformázott szöveget definiál

<q>	páros	2.11	rövidebb sorközi (inline) idézetet definiál
<s>	páros	2.11	nem érvényes/már nem aktuális szöveget definiál
<section>	páros	2.12	a tartalom egy tematikusan összetartozó részét definiálja
<select>	páros	2.14	legördülő listát definiál
<small>	páros	2.11	kisbetűs szöveget definiál
<source>	páros	2.8	multimédia elemet definiál
	páros	2.12	soron belüli (inline) elemet definiál
	páros	2.11	erős kiemelést definiál
<style>	páros	2.2	belső vagy beágyazott stílus definíciót hoz létre
<sub>	páros	2.11	alsó indexet definiál
<sup>	páros	2.11	felső indexet definiál
<table>	páros	2.6	táblázatot definiál
<tbody>	páros	2.6	táblázattörzset definiál
<td>	páros	2.6	táblázat (adat)cellát definiál
<textarea>	páros	2.14	többsoros szövegmezőt definiál
<tfoot>	páros	2.6	táblázat láblécet definiál
<th>	páros	2.6	táblázat fejlécet definiál
<thead>	páros	2.6	táblázatfejet definiál
<title>	páros	2.2	a dokumentum címét definiálja
<tr>	páros	2.6	táblázat sort definiál
<track>	páros	2.8	video feliratozást tartalmaz
<u>	páros	2.11	kommentárt igénylő, aláhúzott szövegrészt definiál
	páros	2.5	számozatlan (felsorolási) listát definiál
<var>	páros	2.11	változó értéket definiál
<video>	páros	2.8	mozgóképet definiál
<wbr>	páratlan	2.11	szótörési lehetőséget definiál

A korábbi szabványból megmaradt, statikus weblapokban alkalmazható alábbi 3 címke nem került tárgyalásra és a fenti abc szerinti felsorolásba:

code számítógépes kód részlet
samp program minta kimenet
base linkek alapértelmezett címe

Az új statikus címkék közül a távol-keleti nyelvekhez tartozó *ruby, rt, rp,* és az írásiránnyal kapcsolatos (közel-keleti) *bdi* és *bdo,* továbbá a *time* (támogatás hiányában valószínűleg törlik a szabvány mostani köréből) maradtak ki az ismertetésből és a fenti, abc szerinti felsorolásból.

Nem kerültek említésre sem a jegyzetben, sem a fenti felsorolásban az akadály-mentesített (WAI-RIA=Web Accessibility Initiative - Accessible Rich Internet Applications) honlapok címkéi.

e) A mintegy 100 statikus weblapi HTML5 címke jelenlegi felhasználhatóságának korlátai:

- az új címkék közül csak a 2011 tavaszától megjelent böngészők értelmezik a *section, nav, article, aside, header* és *footer* címkéket (ez gyakorlatilag az IE 6/7/8-at zárja ki)
- a legújabb böngészőkben sem teljes még a *main, mark, meter* és *wbr* címkék értelmezése (elsősorban az IE 9/10/11 az érintettek)
- a formátumok harca zajlik a *video* és az *audio* funkcióknál

- a *track* címke használhatósága - legutolsóként - a Firefox 31-ben valósult meg, a szöveg szerkesztési nyelve csak *de facto* elfogadott

A HTML5-ön alapuló (európai írásmódú, nem akadálymentesített) statikus web-oldalakhoz tehát mintegy 80 „teljes értékű”, azaz minden böngésző által értelmezett címke használható jelenleg. A szerkezeti és multimedia címkékből további 15 részlegesen értelmezett, de használhatóságuk gyorsan terjed, így több mint 90 címkés repertoárból építkezhetnek a statikus weblapok készítői.

A HTML 4.0/4.01 verziókhoz képest kikerül a szabványból a következő 15 db címke: *acronim, applet, basefont, big, center, dir* (csak alakilag egyezik a *dir* jellemzővel), *font, frameset, frame, isindex, menu, noframes, strike, tt, xmp*. A HTML5 böngészők a visszafelé kompatibilitás jegyében a jövőben is értelmezni tudják őket, de további alkalmazásuk nem javasolt, elavultnak (*obsolete* = elavult) tekintendők.

Megjegyzés: A korábbi (X)HTML-verziók fejlesztésekor kevésbé megengedőek voltak a szabványból kikerült címkékkel szemben, a *deprecated* (helytelen) kategóriába sorolták őket, melyeket az újabb böngészőknek már nem kellett felismerniük.

Az új címkék bevezetésével és a régiek számának fenti csökkentésével a statikus HTML-címkék száma lényegében változatlan maradt.

2.16.2. A HTML címkéknek lehetnek egy adott alkalmazásban kötelező vagy opcionális **jellemzőik** (akár több is) és a jellemzőknek **értékeik**.

A tárgyalt, kötelezően jellemzővel/értékkel ellátott alkalmazások:

<abbr title=”rövidítés kifejtése”> rövidítés </abbr>
megjelenő szöveg v. kép
megjelenő szöveg v. kép
<area href=”……” shape=”……” coords=”……”>
megjelenő szöveg
**
<audio src=”……”>
<datalist id=”…..”>…….option……</datalist>
<dfn title=”meghatározás”> definiálandó szó </dfn>
<iframe src=”……”>
<img src=”……”>
<input type=”……” name=”……”>
<link rel=”stylesheet” href=”fájlnév.css”>
<map id=”……”>……..</map>
<meta charset=”……”>
<meter min=”….” max=”….” value=”….”>
<option value=”……”>
<param name=”……” value=”……”>
<select name=”…..”>…….option……</select>
<source src=”….”>
<textarea name=”……”>……</textarea>
<optgroup label=”……”>
<video src=”……”>

Erősen ajánlottan jellemzővel/értékkel ellátandó alkalmazások:

<audio src=”……” controls preload”……”>
<iframe src=”…..” width=”….” height=”….”>
<img src=”…..” width=”….” height=”….” alt=”…..” title=”…..”>
<meta name=”description” content=”tartalmi összefoglaló”>
<meta name=”keywords” content=”keresőszavak”>
<video src=”……”width=”…..” height=”…..” poster=”……” controls preload=”……”>

2.16.3. A jellemzők lehetnek általánosak (*global attributes*), melyek (majdnem) minden címkével használhatók, vagy specifikusak, melyek adott néhány címkénél használhatók.

Általános jellemzők az alaptulajdonságokat definiáló *id, title, class* és *style* (csak formailag azonos a *style* HTML címkével!), és a nyelvi tulajdonságokat definiáló *lang* és *dir* (a *dir* az írás irányára vonatkozik, ezzel nem foglalkoztunk). Az alaptulajdonságokat definiáló *id, title, class* és *style* jellemzők nem használhatók a *head, html, meta, style* és *title* címkékkel. (A *class* és *style* jellemzőkről a CSS részben lesz szó.)

A leggyakrabban előforduló **specifikus jellemzők** az adott HTML címkéknél kerültek ismertetésre. A HTML5 szerint nem szabványos néhány korábban gyakran alkalmazott címke/jellemző kombináció, pl. az *img* címkéhez a *name* és *align* jellemzők, a *body* címkéhez a *background* és *bgcolor* jellemzők, a *table* címkéhez az *align, bgcolor, border, cellpadding, cellspacing* és *width* jellemzők, a *td* és *th* címkékhez az *align, bgcolor, height, width* és *nowrap* jellemzők, az *a* címkéhez a *name* és *charset* jellemzők, stb. Ezeket sem megtanulni, sem alkalmazni már nem célszerű – bár a böngészők a visszafelé kompatibilitás jegyében továbbra is értelmezni tudják őket.

Megjegyzés: A csak az akadálymentesített honlapoknál használható jellemzőket nem említi a jegyzet, ill. a fenti felsorolás.

2.16.4. Az érték nélküli jellemzők – a kötelező értékűekkel való egységes forma megvalósítása érdekében – a jellemzőt értékként megismételve is kódolhatók, pl.:

download=”`download`***”, controls=”***`controls`***”, loop=”***`loop`***”,*** `stb.`

2.16.5. Minden weblap kódolása bármelyik böngészővel megnézhető. Olyan weblapokat, melyek nagyon tetszenek vagy nem tetszenek, nagyon hasznos és tanulságos ilyen szempontból is tanulmányozni.

Három momentum emlékeztetőnek a **korábbi HTML-verzió**jú oldalak forráskódjának megtekintéséhez:

a) A kötelező *doctype* választás miatt a dokumentumtípus meghatározást az előző szabványokban pl. így kellett megadni:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Transitional helyett lehetett *Strict, HTML* helyett *XHTML, 4.01* helyett *1.0* vagy *1.1,* stb. a dokumentumtípus választástól függően.

Ezek helyett van a HTML5-ben a sokkal egyszerűbb és minden böngésző által egyformán értelmezett *<!doctype html >* definíció.

b) Az előző szabványokban a karakterkészlet definiálása szintén terjengősebb volt:

```
<meta http-equiv="content-type" content="text/html;
charset=iso-8859-2">
```

E helyett van az egyszerűbb <*meta charset=”utf-8”*> (vagy *iso-8859-2*) definíció.

c) A páratlan HTML címkék lezárását a régebbi szabványokban /> jellel, nem pedig egyszerűen > –vel kellett végrehajtani.

d) A HTML5 szerint már elavult címkéket és címke/jellemző kombinációkat lásd a 2.16.1. és 2.16.3. pontokban.

2.16.5. A képernyő tulajdonságok megadhatók HTML-ben a *link* címkével külső CSS stíluslap csatolásaként vagy CSS kódként (ezzel a CSS részben foglalkozunk).

A *media* jellemző a kimeneti (megjelenítő) eszközre vonatkozó feltétel(ek) leírására használatos. Két logikai állapot közül választ – vagy egyezik a megjelenítő eszköz a megadottal, és akkor érvényes a hozzárendelt stílus, vagy nem egyezik, és akkor az a stílus nem érvényesül. A *media* jellemzőben felsorolásként egyszerre több média paraméter is definiálható, a felsorolásban a vessző a „logikai vagy”-ot (tehát a több feltétel közül legalább egynek a teljesülését), az *and* a „logikai és”-t (tehát a több feltétel egyidejű teljesülését) jelenti.

Példa egy 1100 *px* és 1300 *px* közötti szélességű képernyő esetén érvényes, a …….**.css** fájlban megadott megjelenítési stílusnak a definiálására:

```
<link rel="stylesheet" media="screen and (min-width:1100px)
          and (max-width:1300 px )" href= "…….css">
```

A képernyős megjelenítést a *screen* érték definiálja, melynek paraméterei még tovább pontosíthatók, mint pl. *width, height, device-width, device-height, orientation* (portrait, landscape = álló, fekvő), *aspect-ratio* (4/3, 3/2), *device-aspect-ratio* (16/9, 1280/720), *color* (színenkénti bitek száma), *monochrome* (pixelenkénti bitek száma).

2.16.6. Egy webhely kódjai „tisztaságának”, azaz szabványosságának vagy „érvényességének” ellenőrzését *validálás*nak nevezik. A W3C-ben 2011. végén munkacsoport jött létre a HTML5 szabvány szerinti validálás kidolgozására. Rajtuk kívül egyes böngészőkkel vagy külön erre a célra készített szoftverekkel ellenőrizhető a kódolás szabványossága, azonban ezek a HTML5 esetében – tekintettel a szabvány lezáratlanságára - még nem teljesen kiforrottak.

3. CSS3

A CSS nyelvtanának és alkalmazásának tárgyalása előtt a formázási rész öt általános ismertetővel kezdődik.

3.1. Kitöltőszöveg és kitöltőkép

a) A nyomdászatban és az informatikában a betűtípusok, a tipográfia és az elrendezés bemutatására használnak kitöltő- vagy vakszöveget, más szóval szövegutánzatot (*filler, dummy text, placeholder text*). Az ilyen szövegnek nincsen önmagában értelmes jelentése, egyedüli célja, hogy az elrendezésre lehessen koncentrálni úgy, hogy a tartalom ne vonja el a figyelmet a megjelenítésről.

A leggyakrabban használt vakszöveg a ***lorem ipsum*** (röviden: ***lipsum***), mely egy latint utánzó összefüggő szöveg. Eredete a XVI. századra nyúlik vissza, amikor egy ismeretlen nyomdász eltorzított latin szöveget csinált a különböző nyomdai elrendezések bemutatására. Nemrég sikerült kideríteni, hogy a szövegrészt Cicero Kr. e. 45-ben írt *De finibus bonorum et malorum* („A legfőbb jóról és rosszról”) című (az etika elméletét tanulmányozó) műve néhány bekezdésének véletlenszerűen összevágott szavaiból alakították ki. Általában az egyes szavaknak sincs jelentésük, mivel értelmes szavakat vágtak ketté, máskor az egymás után következő szavak szótagjait egyesítették.

Manapság a *lipsum* szövegnek sok változata létezik, néhányuk olyan betűket (például a *k, w, z*) is tartalmaz, melyek hiányoznak a latinban. Ha hosszabb vakszövegre van szükség, a *lipsum* tetszőleges számú ismétlésével szokás a szöveg hosszát megnövelni. A *lipsum*-ok elterjedt használatának oka az is, hogy többé-kevésbé arányosan oszlanak el a szövegben a betűk és a szavak. Az itt használt változat az alábbi:

„Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare quam littera gothica, quam nunc putamus parum claram, anteposuerit litterarum formas humanitatis per seacula quarta decima et quinta decima. Eodem modo typi, qui nunc nobis videntur parum clari, fiant sollemnes in futurum.”

A számtalan lipsum-generátor programmal bárki elő is állíthat magának kitöltőszöveget. A *www.lipsum.com* ingyenes webhely pl. bekezdéseket vagy listákat generál a megadott típusnak és elemszámnak megfelelően az alábbiak szerint:

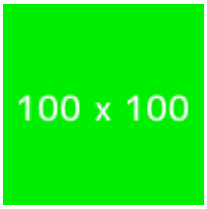
- *Lorem ipsum dolor sit amet, consectetur adipiscing elit.*
- *Phasellus non massa lorem, vel rutrum lectus.*
- *Praesent in sem vel erat dignissim interdum nec et velit.*
- *Vivamus lobortis porttitor nisi, sed interdum dolor facilisis quis.*

- *Nullam ut libero eu est commodo ornare et venenatis nibh.*
- *Nunc tincidunt metus eget lectus tristique auctor varius felis varius.*
- *Sed tempor felis eu lacus tempor ac ornare diam ultrices.*
- *Sed at sem sagittis lacus eleifend scelerisque.*
- *Integer sed elit id dui iaculis luctus.*
- *Suspendisse dictum nibh nec orci ornare non vulputate dui tincidunt.*

- *Suspendisse varius dapibus dui, in dapibus velit ultrices non.*
- *Nullam molestie bibendum urna, non vulputate ipsum dapibus eget.*
- *Vivamus vel sapien velit, et mollis turpis.*
- *Donec sit amet massa ac nulla placerat fermentum.*

Generated 3 paragraphs, 110 words, 795 bytes of Lorem Ipsum

b) Hasonló megfontolásból lehet a webszerkesztés során **kitöltőképeket** is alkalmazni. Erre nincsen általános gyakorlat, egy lehetséges megoldás a *http://dummyimage.com* ingyenes webhely, ahonnan a kívánt képpontok számát beírva (és igény esetén azt rövid szöveggel, azonosítóval, és/vagy színekkel is ellátva) semleges kitöltőképek tölthetők le. Pl. ha egy 320x160 pixeles háttérképet és egy 100x100 pixeles képet szándékozunk majd a weboldalunkon elhelyezni, a dokumentum tervezése során ehhez hasonló kitöltőképekkel alakíthatók ki az elrendezési variánsok:



3.2. Képformátumok

A **képalkotás** módja lehet vektorgrafikus vagy rasztergrafikus (bitkép, bitmapkép, bittérkép, raszterkép).

A vektorképek részleteit matematikai képlettel megadható elemek (egyenesek, görbék) adják, tehát a matematikai paraméterek változtatásával minőségromlás nélkül átméretezhetők. A vektorgrafikába (*SVG=Scalable Vector Graphics*) való bevezetést a 4. fejezet tartalmazza.

A raszterképek egymáshoz illeszkedő, sorokban és oszlopokban elhelyezkedő négyzet alakú képpontokból, angolul *pixel*-ekből (*pixel=picture element*=képpont) állnak. A pixelek a kép tovább már nem osztható alapelemei, felületük nem tartalmazhat tónusátmenetet. A kép **felbontását** a pixelek száma szabja meg, mely a vízszintes és függőleges képpontok számával adható meg (pl. 300x300 pixel). A kép addig nagyítható, ameddig az egyes képpontok nem válnak láthatóvá (pixelesedés), hanem folyamatos képet alkotnak. Általános szabály a képek felbontási követelményére: nyomtatásra 300 ppi, képernyőre 72 ppi (pixels per inch) felbontás.

A digitális képek fő jellemzői a fájl-formátum, a színmód, a színtér és a bitmélység.

Az alkalmazandó **fájl-formátumot** az szabja meg, hogy milyen céleszközre szánunk egy képet, mert a programnyelvek ill. eszközök tulajdonságai eltérőek. A sok képfájl-formátum közül weboldalaknál általánosan a JPEG, GIF és a PNG használható, a Google újabban a WebP elterjesztésére törekszik.

JPEG - kiejtése „jépeg" - (Joint Photographic Experts Group = Egyesült Fényképész-szakértők Csoportja) fájl-formátum veszteséges tömörítést alkalmaz - minél erősebb tömörítést használunk, annál több adat vész el. Max. 16,7 millió szín ábrázolására alkalmas, fotóknál és folytonos színátmeneteket tartalmazó képeknél a legcélszerűbb a használata - kiterjedt, egyszínű területeket tartalmazó képeknél nem érdemes JPEG-et használni. Nem támogatja az átlátszóságot és az animációt.

GIF – kiejtése „gif" - (Graphics Interchange Format = grafika/kép csereformátum) fájl-formátum vonalrajzok, vektorgrafikák, egyszerű emblémák, feliratok, kevés színt, éles határokkal elválasztott egyszínű felületeket tartalmazó képek használatakor a legalkalmasabb. Hatékony veszteségmentes tömörítést alkalmaz, támogatja az animációt (egy fájlban több képkocka tárolását) és az átlátszóságot (egy képpont vagy teljesen átlátszó vagy teljesen átlátszatlan), rendelkezik váltottsoros (interlaced) üzemmóddal (letöltéskor fokozatosan finomodva jelenik meg a kép, mert a böngésző egy kisebb felbontású képpel indít).

Indexelt színmódban, 256-színű színpalettából (*color lookup table*) választhatók a színek – ha egy képet színpalettássá alakítunk, és az eredeti szín nem létezik a palettán, akkor a hozzá legközelebbi színbe megy át, tehát nem pontosan színtartó. Ugyanakkor a 256 színből a képben használandó színek száma le is csökkenthető csak a ténylegesen felhasználtakra, így két-három színből gombokat, ikonokat lehet nagyon kis méretű, gyorsan letöltődő képfájl-okban létrehozni.

PNG - kiejtése „ping" - (Portable Network Graphic = hordozható hálózati grafika) fájl-formátumot kimondottan web-es alkalmazásra fejlesztette ki a W3C. Veszteségmentes tömörítést alkalmaz, lehetővé teszi az áttetszőséget és a váltottsoros megjelenítést, de nem támogatja az animációt.

PNG-8: 8 bit-es változat, az alkalmazhatósága hasonló a GIF-hez, 256 színt, bináris átlátszóságot támogat.

PNG-24: 24 bit-es változat, hasonló az alkalmazhatósága a JPEG-hez, de a veszteségmentes tömörítés miatt nagyobb fájlméret adódik mint az azonos JPEG képnél, viszont nincsen tömörítési képminőségromlás, és 256-fokozatú átlátszóságot támogat.

Mozilla vezetésével *MNG* = Multiple-image Network Graphic, majd *APNG* = Animated Network Graphic formátumok kerültek kidolgozásra a PNG-alapú animáció megvalósítására, azonban böngészőkben való támogatottságuk és szabványosításuk még (?) nem következett be – az APNG-t a Firefox-on kívül csak a Safari 8 / iOS Safari 8 értelmezik.

A JPEG web-es környezetben való leváltásán dolgozik a Google a **WebP** - kiejtése „veppi"- szabvánnyal (egyetlen képkocka WebM codec-el kódolva), mellyel kb. 40%-al kisebb fájlméret elérése a cél. Bár a Google a gmail-ben, Picasa-ban, a Chrome és Android böngészőiben is támogatja, csak az új Opera böngészők csatlakoztak a támogatásához.

A színmóddal, színtérrel és bitmélységgel a *Színek* rész foglalkozik.

3.3. Színek

A színmód adja meg, hogy egy adott színárnyalat a színek milyen alaphalmazából és milyen módon épül fel. Szín minden képponthoz (képpont = pixel = *picture element*) rendelhető.

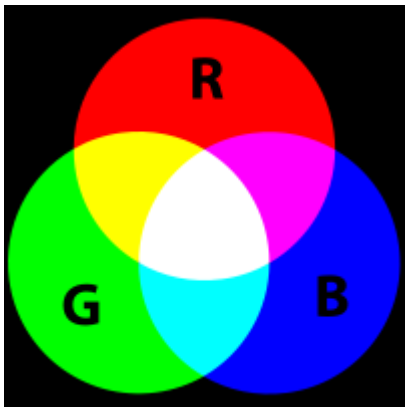
Az információ alapegysége a *bit* (bit =*binary digit*=kettes számrendszerbeli számjegy), mely 0 vagy 1 értéket vehet fel. 1 bit-es színmód a Bitmap színmód (nem összekeverendő a bitmap képformátummal), melyben egy képpont vagy fekete vagy fehér.

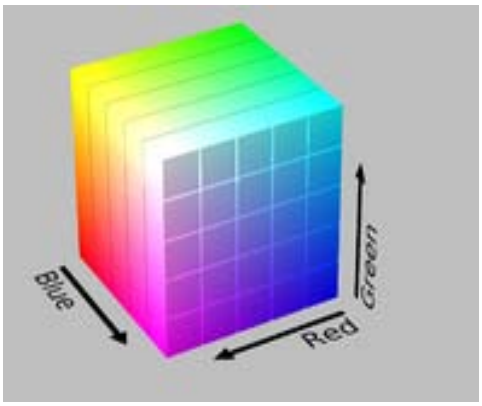
A számítógépben az információfeldolgozás alapegysége a bájt (*byte*). 1 bájt = 8 bit, mely 2^8 = 256 értéket vehet fel. 8 bit-es színmód a Grayscale (szürkeárnyalat) színmód, melyben a fekete és fehér színekből a fehértől a feketéig 256-féle szürkeárnyalat állítható elő.

3.3.1. RGB(A)

A sok létező színes színmód közül a weblapok megjelenítésére döntően a **24-bites (3x8 bites) RGB** színmódot használják, mely három alapszín (Red = vörös, Green = zöld, Blue = kék) keveréséből állít elő minden színárnyalatot.

A színkeverés három alapszín különböző arányú összegzésével működik, ezért ezt *additív* (összegző, összeadó) színkeverésnek nevezik. A bal oldali kép síkban, a jobb oldali térben illusztrálja a 3 alapszín összeadásából kialakuló színeket.





Mivel ebben a színmódban az alkalmazott RGB-színek alapszínenként 1 bájt-tal (8 bit-el), azaz összesen 3 x 8 = 24 bit-el adhatók meg képpontonként, 256^3 = 16 777 216 szín állítható így elő (ezt hívják 24 bit-es True Color-nak). Az emberi szem kb. 10 millió színárnyalat megkülönböztetésére képes, ezért (speciális alkalmazásoktól eltekintve) nem érdemes az alapszínek még finomabb keverését előírni.

Nem minden megjelenítőeszköz képes visszaadni egy színmód minden lehetséges színkombinációját, ezért egy színmódon belül bizonyos színkombináció-csoportokat hoztak létre – ezek a **színterek**, más néven színpaletták, gamutok vagy színprofilok. A monitorokra, asztali nyomtatókra és az internetes alkalmazásokra a Hewlett Packard és a Microsoft fejlesztette ki 1996-ban az sRGB (standard RGB) színteret, melyet a W3C is elfogadott.

Egy adott színárnyalatban a bekevert alapszínek intenzitása **tízes számrendszerben** alapszínenként 0-255 közötti pozitív egész számokkal adható meg (2^8 = 256 érték), a három alapszínből előállított tetszőleges színárnyalat tehát három, a vörös, zöld és kék intenzitását megadó, 0-255 közötti számmal definiálható (a fenti sorrend kötelező). A fekete pl. (0, 0, 0),

a fehér (255, 255, 255), a vörös (255, 0, 0), a zöld (0, 255, 0), a kék (0, 0, 255), a sárga (255, 255, 0), a cián (0, 255, 255), a bíbor (255, 0, 255), stb. Ha a három alapszín intenzitási értéke egyforma, eredményként semleges színt – fehéret, feketét, vagy valamilyen színezet nélküli szürkeárnyalatot – kapunk.

Hexadecimális (tizenhatos számrendszerbeli, röviden „hexa”) megadás esetén 0 és 9 között számokkal, 10 és 15 között az **a** és **f** közötti betűkkel történik az alapszínek intenzitásának definiálása. A 6 karakterből álló színkódban az érték elé kötelezően # jelet (hash = oktotorp, kettőskereszt) kell tenni. Az első két karakter a piros, a második kettő a zöld, az utolsó két karakter a kék alapszín intenzitását határozza meg (*hex triplet*). A fentiek alapján a fekete #000000, a fehér #ffffff, ha mindhárom érték ugyanaz (páronként azonos karakterekből állnak), akkor a fekete és fehér közötti valamelyik szürke árnyalat jön létre, a #ff0000 a vörös, #008000 a zöld, a #0000ff a kék, stb. Ha az egyes színösszetevők (két-két) azonos számból vagy betűből állnak, elég a párosokból csak az egyik karaktert megadni, így 3 karakteres rövid kód (*short notation*) is használható, pl. a #ffbb00 színkód megadható #fb0 – ként.

Tízes számrendszerbeni (0 – 255) értéket hexadecimálisra úgy lehet átszámítani (mint a 10-es számrendszerből a tizenhatosba való átmenet esetén), hogy 16-al osztani kell a tízes számrendszerbeli értéket, az egészszámú részéhez hozzárendeljük a hexa értéket, a maradékhoz pedig ismét a 16-os számrendszerbeli értéket. Pl. ha egy alapszín tizes számrendszerbeli intenzitása 201, akkor 201/16=12,5625; a 12-nek **c** felel meg, a maradék 201-12x16=9, tehát hexá-ban c9 lesz.

Ilyen átszámításra a gyakorlatban nincsen szükség – egy tetszőleges színárnyalat kódját egy képszerkesztő program pipetta eszközével vagy a legtöbb webszerkesztő program színpalettájából közvetlenül lehet leolvasni abban a számrendszerben, amiben kívánjuk. Különben is a webbiztos színekre ill. a névvel megadott színekre (lásd a továbbiakban) kész megoldás található.

A 00, 33, 66, 99, CC és FF karakterpárok kombinációiból előállítható színeket hívják **webbiztos színek**nek, melyek árnyalatai és hexa-kódjai a következő táblázatban láthatók.

A webbiztos színek (*web-safe colors*, vagy *browser-safe colors*) operációs rendszertől és megjelenítő eszköztől függetlenül azonos színárnyalatként jelennek meg. Ha a három alapszín fenti 6 – 6 – 6 árnyalatát vesszük, 6^3 = 216 szín hozható létre. (Régen a display-ek is 216 színt tudtak megjeleníteni.)

#000000	#330000	#660000	#990000	#CC0000	#FF0000
#000033	#330033	#660033	#990033	#CC0033	#FF0033
#000066	#330066	#660066	#990066	#CC0066	#FF0066
#000099	#330099	#660099	#990099	#CC0099	#FF0099
#0000CC	#3300CC	#6600CC	#9900CC	#CC00CC	#FF00CC
#0000FF	#3300FF	#6600FF	#9900FF	#CC00FF	#FF00FF
#003300	#333300	#663300	#993300	#CC3300	#FF3300
#003333	#333333	#663333	#993333	#CC3333	#FF3333
#003366	#333366	#663366	#993366	#CC3366	#FF3366
#003399	#333399	#663399	#993399	#CC3399	#FF3399
#0033CC	#3333CC	#6633CC	#9933CC	#CC33CC	#FF33CC
#0033FF	#3333FF	#6633FF	#9933FF	#CC33FF	#FF33FF

#006600	#336600	#666600	#996600	#CC6600	#FF6600
#006633	#336633	#666633	#996633	#CC6633	#FF6633
#006666	#336666	#666666	#996666	#CC6666	#FF6666
#006699	#336699	#666699	#996699	#CC6699	#FF6699
#0066CC	#3366CC	#6666CC	#9966CC	#CC66CC	#FF66CC
#0066FF	#3366FF	#6666FF	#9966FF	#CC66FF	#FF66FF
#009900	#339900	#669900	#999900	#CC9900	#FF9900
#009933	#339933	#669933	#999933	#CC9933	#FF9933
#009966	#339966	#669966	#999966	#CC9966	#FF9966
#009999	#339999	#669999	#999999	#CC9999	#FF9999
#0099CC	#3399CC	#6699CC	#9999CC	#CC99CC	#FF99CC
#0099FF	#3399FF	#6699FF	#9999FF	#CC99FF	#FF99FF
#00CC00	#33CC00	#66CC00	#99CC00	#CCCC00	#FFCC00
#00CC33	#33CC33	#66CC33	#99CC33	#CCCC33	#FFCC33
#00CC66	#33CC66	#66CC66	#99CC66	#CCCC66	#FFCC66
#00CC99	#33CC99	#66CC99	#99CC99	#CCCC99	#FFCC99
#00CCCC	#33CCCC	#66CCCC	#99CCCC	#CCCCCC	#FFCCCC
#00CCFF	#33CCFF	#66CCFF	#99CCFF	#CCCCFF	#FFCCFF
#00FF00	#33FF00	#66FF00	#99FF00	#CCFF00	#FFFF00
#00FF33	#33FF33	#66FF33	#99FF33	#CCFF33	#FFFF33
#00FF66	#33FF66	#66FF66	#99FF66	#CCFF66	#FFFF66
#00FF99	#33FF99	#66FF99	#99FF99	#CCFF99	#FFFF99
#00FFCC	#33FFCC	#66FFCC	#99FFCC	#CCFFCC	#FFFFCC
#00FFFF	#33FFFF	#66FFFF	#99FFFF	#CCFFFF	#FFFFFF

A korszerű böngészők elég nagy pontossággal jelenítik meg a színárnyalatokat, így ma már a webbiztos színeken kívül is használható egy lényegesen nagyobb színválaszték.

Százalékos arány esetén a vörös, zöld és kék (kötelezően ebben a sorrendben) keverési arányát határozzuk meg. A százalékok azt fejezik ki, hogy az adott alapszín 100% - ához képest mennyit keverünk bele – tehát az alapszínek százalékos összegének nem kell kiadnia a 100%-ot!

Százalékos megadás esetén az értékek természetesen 0% és 100% között mozoghatnak. A fehér színben pl. mindhárom alapszín teljes mértékben szerepel, így értéke (100%, 100%, 100%).

16 olyan szín van – ezeket **alap(vető) színnevek**nek (*basic color keyword*) is nevezik - melyek az (angol) nevükkel is megadhatók, és nevüket kívülről szokás tudni. Ezek ABC-sorrendben:

szín	**hexa-kód**	**szín**	**hexa-kód**	**szín**	**hexa-kód**	**szín**	**hexa-kód**
aqua	#00FFFF	gray	#808080	navy	#000080	silver	#C0C0C0
black	#000000	green	#008000	olive	#808000	teal	#008080
blue	#0000FF	lime	#00FF00	purple	#800080	white	#FFFFFF
fuchsia	#FF00FF	maroon	#800000	red	#FF0000	yellow	#FFFF00

A magyar megfelelőjük:

aqua = akvamarinkék	gray = szürke	navy = tengerészkék	silver = ezüstszürke
black = fekete	green = zöld	olive = olajzöld	teal = pávakék
blue = kék	lime = világoszöld	purple = lila	white = fehér
fuchsia = mályva	maroon = gesztenyebarna	red = vörös	yellow = sárga

További 124 színnek van a W3C által jóváhagyott, nem csak színkóddal de névvel is megadható értéke – ezeket **kiterjesztett színnevek**nek (*extended color keyword*) nevezik - de közülük csak a gyakran használt, kedvenc színek megnevezéseit szokás kívülről tudni. Ez a 124 szín ABC-sorrendben:

színárnyalatok	**színnevek**	**hexa**	**decimális**
	aliceblue	#f0f8ff	240,248,255
	antiquewhite	#faebd7	250,235,215
	aquamarine	#7fffd4	127,255,212
	azure	#f0ffff	240,255,255
	beige	#f5f5dc	245,245,220
	bisque	#ffe4c4	255,228,196
	blanchedalmond	#ffebcd	255,235,205
	blueviolet	#8a2be2	138,43,226
	brown	#a52a2a	165,42,42
	burlywood	#deb887	222,184,135
	cadetblue	#5f9ea0	95,158,160
	chartreuse	#7fff00	127,255,0
	chocolate	#d2691e	210,105,30
	coral	#ff7f50	255,127,80
	cornflowerblue	#6495ed	100,149,237
	cornsilk	#fff8dc	255,248,220
	crimson	#dc143c	220,20,60
	cyan	#00ffff	0,255,255
	darkblue	#00008b	0,0,139
	darkcyan	#008b8b	0,139,139
	darkgoldenrod	#b8860b	184,134,11
	darkgray	#a9a9a9	169,169,169
	darkgreen	#006400	0,100,0
	darkkhaki	#bdb76b	189,183,107
	darkmagenta	#8b008b	139,0,139
	darkolivegreen	#556b2f	85,107,47
	darkorange	#ff8c00	255,140,0
	darkorchid	#9932cc	153,50,204
	darkred	#8b0000	139,0,0
	darksalmon	#e9967a	233,150,122
	darkseagreen	#8fbc8f	143,188,143

	darkslateblue	#483d8b	72,61,139
	darkslategray	#2f4f4f	47,79,79
	darkturquoise	#00ced1	0,206,209
	darkviolet	#9400d3	148,0,211
	deeppink	#ff1493	255,20,147
	deepskyblue	#00bfff	0,191,255
	dimgray	#696969	105,105,105
	dodgerblue	#1e90ff	30,144,255
	firebrick	#b22222	178,34,34
	floralwhite	#fffaf0	255,250,240
	forestgreen	#228b22	34,139,34
	gainsboro	#dcdcdc	220,220,220
	ghostwhite	#f8f8ff	248,248,255
	gold	#ffd700	255,215,0
	goldenrod	#daa520	218,165,32
	greenyellow	#adff2f	173,255,47
	honeydew	#f0fff0	240,255,240
	hotpink	#ff69b4	255,105,180
	indianred	#cd5c5c	205,92,92
	indigo	#4b0082	75,0,130
	ivory	#fffff0	255,255,240
	khaki	#f0e68c	240,230,140
	lavender	#e6e6fa	230,230,250
	lavenderblush	#fff0f5	255,240,245
	lawngreen	#7cfc00	124,252,0
	lemonchiffon	#fffacd	255,250,205
	lightblue	#add8e6	173,216,230
	lightcoral	#f08080	240,128,128
	lightcyan	#e0ffff	224,255,255
	lightgoldenrodyellow	#fafad2	250,250,210
	lightgray	#d3d3d3	211,211,211
	lightgreen	#90ee90	144,238,144
	lightpink	#ffb6c1	255,182,193
	lightsalmon	#ffa07a	255,160,122
	lightseagreen	#20b2aa	32,178,170
	lightskyblue	#87cefa	135,206,250
	lightslategray	#778899	119,136,153
	lightsteelblue	#b0c4de	176,196,222
	lightyellow	#ffffe0	255,255,224
	limegreen	#32cd32	50,205,50
	linen	#faf0e6	250,240,230

	magenta	#ff00ff	255,0,255
	mediumaquamarine	#66cdaa	102,205,170
	mediumblue	#0000cd	0,0,205
	mediumorchid	#ba55d3	186,85,211
	mediumpurple	#9370db	147,112,219
	mediumseagreen	#3cb371	60,179,113
	mediumslateblue	#7b68ee	123,104,238
	mediumspringgreen	#00fa9a	0,250,154
	mediumturquoise	#48d1cc	72,209,204
	mediumvioletred	#c71585	199,21,133
	midnightblue	#191970	25,25,112
	mintcream	#f5fffa	245,255,250
	mistyrose	#ffe4e1	255,228,225
	moccasin	#ffe4b5	255,228,181
	navajowhite	#ffdead	255,222,173
	oldlace	#fdf5e6	253,245,230
	olivedrab	#6b8e23	107,142,35
	orange	#ffa500	255,165,0
	orangered	#ff4500	255,69,0
	orchid	#da70d6	218,112,214
	palegoldenrod	#eee8aa	238,232,170
	palegreen	#98fb98	152,251,152
	paleturquoise	#afeeee	175,238,238
	palevioletred	#db7093	219,112,147
	papayawhip	#ffefd5	255,239,213
	peachpuff	#ffdab9	255,218,185
	peru	#cd853f	205,133,63
	pink	#ffc0cb	255,192,203
	plum	#dda0dd	221,160,221
	powderblue	#b0e0e6	176,224,230
	rosybrown	#bc8f8f	188,143,143
	royalblue	#4169e1	65,105,225
	saddlebrown	#8b4513	139,69,19
	salmon	#fa8072	250,128,114
	sandybrown	#f4a460	244,164,96
	seagreen	#2e8b57	46,139,87
	seashell	#fff5ee	255,245,238
	sienna	#a0522d	160,82,45
	skyblue	#87ceeb	135,206,235
	slateblue	#6a5acd	106,90,205
	slategray	#708090	112,128,144

	snow	#fffafa	255,250,250
	springgreen	#00ff7f	0,255,127
	steelblue	#4682b4	70,130,180
	tan	#d2b48c	210,180,140
	thistle	#d8bfd8	216,191,216
	tomato	#ff6347	255,99,71
	turquoise	#40e0d0	64,224,208
	violet	#ee82ee	238,130,238
	wheat	#f5deb3	245,222,179
	whitesmoke	#f5f5f5	245,245,245
	yellowgreen	#9acd32	154,205,50

Figyelem! Színnévként megadható a teljesen átlátszó *(transparent)* is – ami végül is csak formailag szín, hiszen színtelen átlátszóságot definiál. A szürke megnevezést pedig (önállóan és szóösszetételben is) *grey* és *gray* írásmódban is elfogadja a szabvány.

Fentiek alapján ugyanaz a vörös szín tehát az alábbi módokon megadható:

hexa	(#ff0000)
rövid hexa	(#f00)
decimális	(255, 0, 0)
százalék	(100%, 0%, 0%)
sRGB név	red

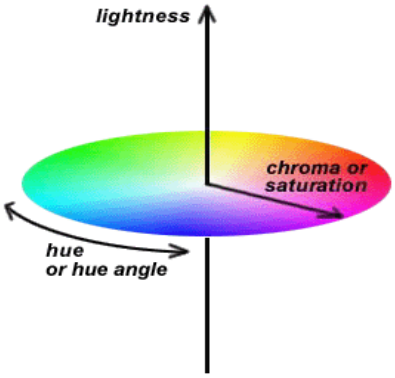
Az RGB kiterjesztésével az **átlátszóság** (*opacity*) is specifikálható. Az RGBA-ban az *A* (alfa) az átlátszóság, mely 0 és 1 közötti értéket vehet fel: 0 a teljesen átlátszó (egyáltalán nem takarja az alatta lévő színt), 1 a teljesen átlátszatlan (teljesen takarja az alatta lévő színt).

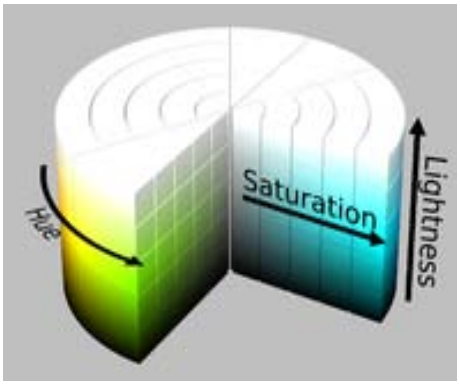
Az RGBA-nak nincsen névvel vagy hexa-ban megadható formája, a szín átlátszósága a decimális vagy százalékos módszer kiegészítésével adható meg, amit tizedesponttal - nem tizedesvesszővel (!) - kell a színek értékei után definiálni.

Például RGBA-ban egy félig átlátszó piros szín értéke (255, 0, 0, 0.5), a teljesen átlátszó „szín” pedig (0, 0, 0, 0) - vagyis a korábban ismertetett *transparent* színnév tulajdonképpen a teljesen átlátszó fekete szín rövid formában való definiálása.

3.3.2. HSL(A)

Az RGB mellett választékbővítésként vezette be a W3C a **HSL színmódot**, melyben egy adott színárnyalatot a H=*hue*=színezet, S=*saturation*=(szín)telítettség, L=*lightness*=világosság/fényesség paraméterekkel definiálnak.





A színezet a (szivárvány színeit ábrázoló) színkörön a vöröshöz képest bezárt szöggel adható meg, tehát definíciószerűen a vörös=0°=360°, a zöld pl. 120°-al, a kék 240°-al írható le, stb. A színtelítettséget és fényességet 0%-100% között százalékosan kell megadni – telítettség esetén a 0% egy szürke árnyalat és 100% a teljes színtelitettség, míg a világosságnál 0% a fekete és 100% a fehér (a „normális” világosság értéke 50%).

Figyelem! A °-ot nem kell mértékegységként megadni, a %-ot viszont igen!

Az alábbi táblázatok bemutatják a színek alakulását a meghatározásukra szolgáló három paraméter értékeinek függvényében. Példaként 30°-onként haladtunk a színkörön 0°-330° között, és minden táblázatban beskáláztuk a vízszintes tengelyen 100% - 0% között a telítettséget, a függőleges tengelyen pedig 0%-100% között a világosságot:

0° vörösök

telítettség	100%	75%	50%	25%	0%
100					
88					
75					
63					
50					
38					
25					
13					
0					

30° vörös-sárgák

telítettség	100%	75%	50%	25%	0%
100					
88					
75					
63					
50					
38					
25					
13					
0					

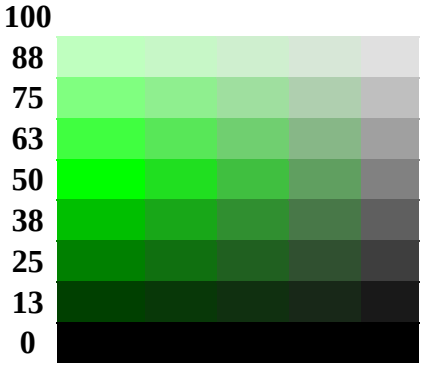
60° sárgák

telítettség	100%	75%	50%	25%	0%
100					
88					
75					
63					
50					
38					
25					
13					
0					

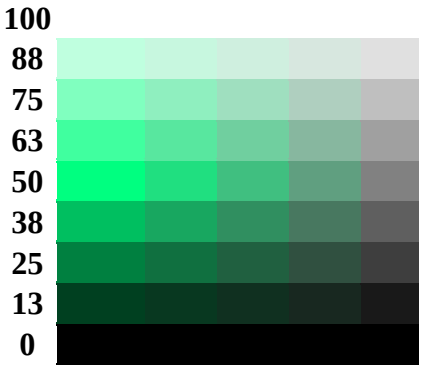
90° sárga-zöldek

telítettség	100%	75%	50%	25%	0%
100					
88					
75					
63					
50					
38					
25					
13					
0					

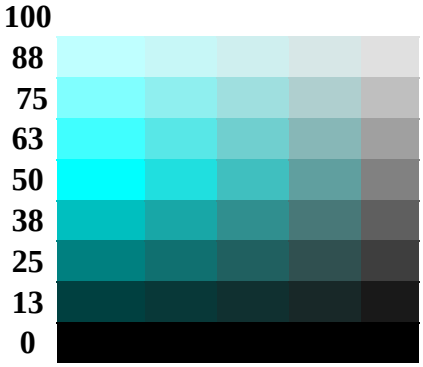
120° zöldek
telítettség
100% 75% 50% 25% 0%



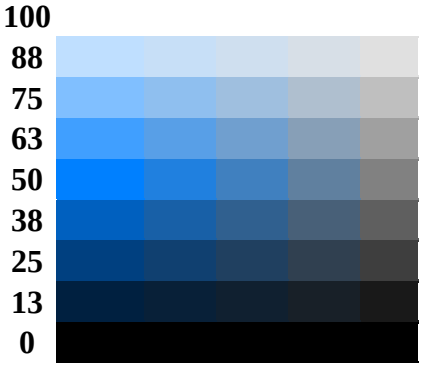
150° zöld-ciánok
telítettség
100% 75% 50% 25% 0%



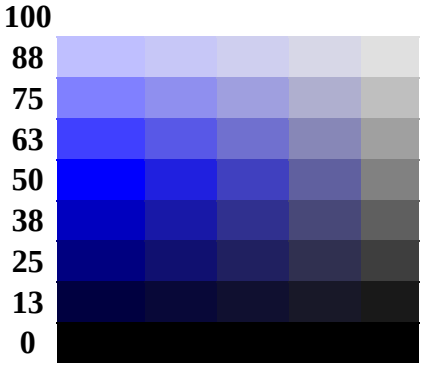
180° ciánok
telítettség
100% 75% 50% 25% 0%



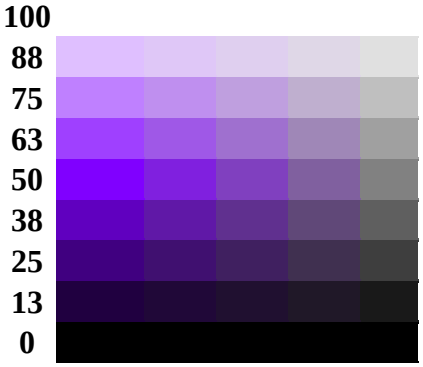
210° cián-kékek
telítettség
100% 75% 50% 25% 0%



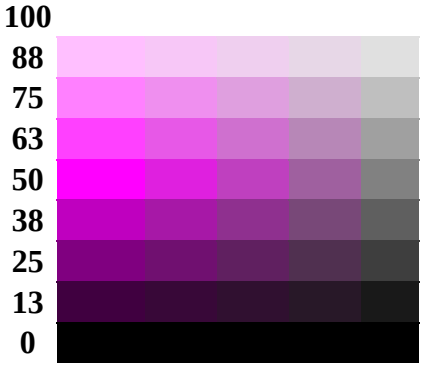
240° kékek
telítettség
100% 75% 50% 25% 0%



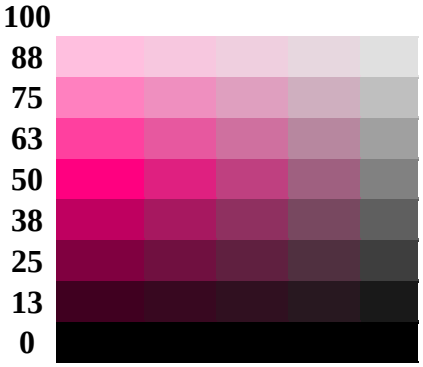
270° kék-bíborok
telítettség
100% 75% 50% 25% 0%



300° bíborok
telítettség
100% 75% 50% 25% 0%



330° bíbor-vörösök
telítettség
100% 75% 50% 25% 0%



HSL-ben a normál vörös szín értéke tehát a fentiek alapján: (0, 100%, 50%)
a világos zöldé: (120, 100%, 75%)
a normál zöldé: (120, 100%, 50%)
a sötét zöldé: (120, 100%, 25%)
stb.

A HSL színmód használatának előnye, hogy

- a színezet a színkörről könnyen megválasztható (ellentétben az RGB elvont színképzésével)
- a normál értékhez viszonyítva szimmetrikusan állítható a világosság/sötétség mértéke
- könnyen átszámítható RGB-re (az RGB ill. HSL térbeli ábrázolásokon látszik a szükséges transzformáció jellege)

Figyelem! Az Internet Explorer-ek csak a 9-es verziótól értelmezik a HSL szín-értékeket.

A HSL kiterjesztésével az átlátszóság (opacity) is specifikálható. Az RGBA-ban látottakkal analóg módon a HSLA a HSL színérték 0 – 1 közötti alfa-értékkel való kibővítésével hozható létre. Ha az alfa-érték = 1, akkor csak külön feltüntettük az átlátszóságot, de a HSL-el megegyező színt definiáltunk, pl. a

(120, 100%, 50%) és (120, 100%, 50%, 1) ugyanazt a normál zöld színt jelenti.

3.3.3. Színharmóniák

A színek objektív jellemzői a színezet, a telítettség és a világosság. Ezeken kívül szubjektív módon is megkülönböztetünk színeket (pl. hidegnek vagy melegnek érezzük őket), és hatásukat az is befolyásolja, hogy milyenek a pillanatnyi érzelmeink, milyen egyéb színekkel együtt jelennek meg, stb.

A konstruktív színelmélet széles körben (többek között festőknek, grafikusoknak, fotósoknak, építészeknek, lakberendezőknek és web-designereknek) támpontokat ad a munka során alkalmazandó színek megválasztásában, melyhez egy tizenkét színű színkört használ:

(1) Három **első rendbeli szín**ből (alapszínből, elsődleges színből, főszínből) indulunk ki, melyek keveréssel nem állíthatók elő (két elemből a harmadik színt nem lehet kikeverni), de az összes többi szín a keverékükből jön létre. Ezek az alapszínek a **sárga, vörös** és **kék** - tehát

nem azonosak az additív színkeverésnél alkalmazott vörös, zöld és kék (RGB) alapszínekkel. A színkör közepét alkotja az első rendbeli színekből álló egyenlő oldalú háromszög.

(2) A három **második rendbeli szín** (mellékszín, másodlagos szín) az alapszínekből keverhető **narancs** (sárga + vörös)**, ibolya** (vörös + kék) és **zöld** (sárga + kék). Az alapszíneket tartalmazó egyenlő oldalú háromszög csúcsait szabályos, egyenlő oldalú hatszöggé kiegészítve az így keletkezett egyenlő szárú háromszögek alkotják a vonatkozó elsődleges színekből létrehozott másodlagos színeket.

(3) A hatszög csúcsait érintő körgyűrűt rajzolva a körgyűrűn a hatszög csúcsai közé eső hat szakasz közepei megfelelnek a szomszédos színek – azaz mindig egy első rendbeli és egy második rendbeli szín – keverékének. Az így kapott hat szín a **harmadik rendbeli** (harmadlagos) színek: a **sárgásnarancs** (sárga + narancs), a **vörösesnarancs** (vörös + narancs), a **vörösesibolya** (vörös + ibolya), a **kékesibolya** (kék + ibolya), a **kékeszöld** (kék + zöld) és a **sárgászöld** (sárga + zöld).

Az ilyen módon megszerkesztett tizenkét osztatú színkör:

Megjegyzés:

- a színkör további színkeverésekkel tovább finomítható lenne egyre inkább közelítve a színek közötti folytonos átmenethez, azonban ez indokolatlanul bonyolultá tenné a használatát
- szín alatt a tiszta, telített színezetet (100% telítettség, 50% világosság) értjük
- a fekete és a fehér nem részei a színkörnek, csak a színek sötétítésére ill. világosítására szolgálnak.

A továbbiakban a szerkesztés eredményeként nyert, az alap- másodlagos és harmadszíneket egy 12-osztású körön egyenletesen elhelyező u. n. *Itten-féle* színkört használjuk:

A színszerkezetek a színek kiválasztott kombinációja, melyek harmonikus vagy kontraszthatásba kerülnek egymással. A színharmónia két vagy több szín közös hatása.

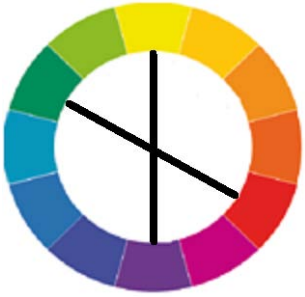
Nevezetes színszerkezetek:

- **akrom(atikus)** azaz “színtelen szín” - nincs hozzákeverve semmilyen más szín. A fehér és a fekete között találhatók a szürke különböző árnyalatai.

- **monokrom(atikus)** azaz “egyszínű, színen belüli” – a színkörből kiválasztott egy szín világosabb ill. sötétebb árnyalatainak kombinációja.

- **analóg** azaz „azonos értékű” színek: a színkörben egymás melletti, egy színcsaládba tartozó, hasonló színek kombinációi, mint pl. a kék - kékeszöld - zöld, vagy a narancs – vöröses narancs – vörös. Ezek a kombinációk is finom hatásúak, de érdekesebbek a monokróm kombinációknál.

- **kiegészítő** (komplementer) színek: a színkörön egymással szemben helyezkednek el, pl. zöld – vörös, vagy sárga – viola, stb. kombinációk. Hatásuk kontrasztos, életteli, feltűnést keltő – a kiegészítő színek felerősítik, kiemelik egymást.



- **osztott kiegészítő** (osztott komplementer, zenei hármashangzat) színek: egy komplementer színpár egyik színét a két szomszédos színére cserélve jönnek létre, egyenlőszárú háromszögek csúcsain helyezkednek el.

- **harmonikus hármashangzatok** színei: egymástól egyforma távolságra, egyenlő oldalú háromszögek csúcsain helyezkednek el. A legharsányabb, „legerősebb” az elsődleges színekből (sárga, vörös, kék) alkotott u.n. „ősi hármashangzat” kombináció, ugyancsak erős a másodlagos színekből (zöld – narancs – viola) alkotott színhármas, „finomabbak” a két, harmadlagos színekből álló (vöröses viola-kékeszöld-sárgás narancs, ill. kékes viola-sárgászöld-vöröses narancs) hármashangzatok (minél több szín alkotja a keveréket - a harmadszínekben már minden alap- és másodszín szerepel - annál finomabb színhatás jön létre).

- **harmonikus négyeshangzatok**: egymástól egyforma távolságra, egy négyzet csúcsain elhelyezkedő színek kombinációja (a négyszög forgatásával három lehetséges változat).

- **zenei négyeshangzatok**: komplementer színpárok mindkét színének a szomszédaira cserélésével jönnek létre, téglalapok csúcsain helyezkednek el (a téglalap forgatásával hat lehetséges változat).

- **ötöshangzatok**: a hármashangzatokat fehérrel és feketével kombinálva ötöshangzatot kapunk. Ilyen például a sárga – vörös – kék – fekete – fehér, vagy a narancs – ibolya – zöld – fehér – fekete.

- **hatoshangzatok: a** színkörbe hatszög alakzatot rajzolva két harmonikus hatoshangzat jön létre, a színkör szerkesztéséből már ismert sárga – narancs – vörös – ibolya – kék – zöld, ill. a hatszög elfordításával keletkező sárgásnarancs – vörösesnarancs – vörösesibolya – kékes ibolya – kékesszöld – sárgászöld.

A színkör színe(zete)i telítettségének és/vagy világosságának változtatásával a kombinációk palettája jelentősen kiszélesedik. A tiszta szín telítettsége szürke árnyalatok, a világossága fehér ill. fekete hozzáadásával változtatható. Például tiszta színek telítettségének és világosságának változtatásával kapott színek:

A telítettséggel és világossággal kapcsolatos szubjektív színhatások:

- tiszta színek: erőteljes, harsány, vidám, ragyogó
- telítetlen színek: fáradt, ködös, fakó, öreg
- világos színek: fályolos, halvány, pasztell, lágy
- sötét színek: haragos, mély, komor, viharos

Ugyancsak szubjektív színhatás az asszociációkon alapuló meleg – hideg színérzet. A tűzhöz, vérhez, naphoz kapcsolódó meleg színek a sárga, a narancs, a vörös és a bíbor. A vízhez, jéghez és éjszakához köthető hideg színek a zöld és a kék. A színkörön a legmelegebb szín a narancsos vörös, és általában meleg színnek nevezik még a sárgát, a sárgásnarancsot, narancsot és a vöröset. A leghidegebb szín a legmelegebbel szemben lévő kék és türkizkék, hidegnek számít még a sárgászöld, zöld, kékeszöld és ibolya.

Ezen a felosztáson belül is minden színnek van hideg és meleg árnyalata. A színkörön egy adott színnek a piros felé eső árnyalata a melegebb, az ellentétes irányba eső árnyalata a hidegebb:

3.4. Értékek és mértékegységek

Az egyes CSS-tulajdonságok értékei az alábbi módokon adhatóak meg:

a) (kulcs)szavakkal (*keyword*): pl. *thin, bold, center, top, auto, disc,* 140 színnek az angol nevükkel való megadása, stb.

b) mértékegység nélküli számmal - ez lehet egész szám (*integer*) vagy valós szám (*real number*), melynél tizedesvessző helyett tizedespontot kell használni. A pozitív és negatív előjel (*sign character*) a tízes számrendszerbeli számérték részét képezi.

c) számmal és mértékegységgel - a szám után közvetlenül, betűköz nélkül jön a mértékegység "0" után opcionális a mértékegység kitétele.

c/1 Abszolút mértékegységek (akkor hasznosak, ha a kimeneti eszköz fizikai tulajdonságai ismertek):

fizikai egységek:	in (inch, *hüvelyk*) = 25,4 mm
	cm (centiméter)
	mm (milliméter)
	pt (point, *pont*) = 1/72 inch
	pica (pica, *ciceró*) = 12pont
referencia-pixel:*	px (picture element, *képelem*) = 1/96 inch

A web-es méretmegadásoknál a „normál" hosszmértékek (hüvelyk, centiméter, milliméter) csak ritkán használatosak.

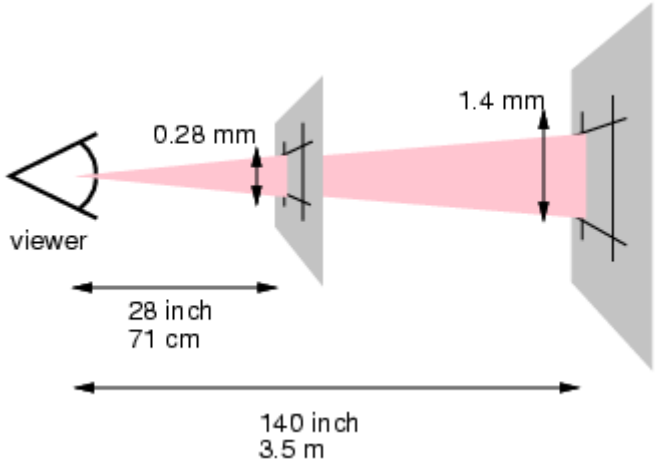
c/2 Relatív mértékegységek (akkor hasznosak, ha különböző kimeneti eszközökre kell a megjelenítést rugalmasan átméretezni):

- százalékos definiálás = a felhasználó által megadott mérethez viszonyított méret
- em (**nem** azonos az *em* HTML-címkével) = egy elem betűméretének aktuális értéke, a felhasználó által megadott betűmérethez viszonyított méret
- rem = a gyökérelem betűmérete

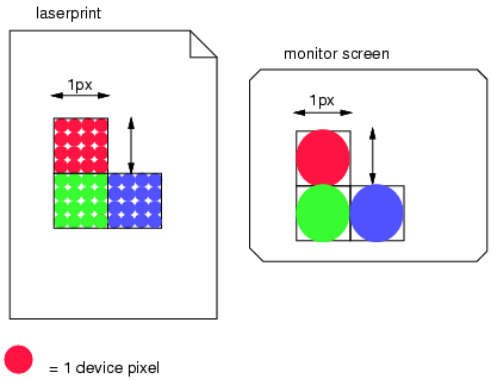
*A *px* a megjelenítő eszköz (leggyakrabban egy monitor vagy egy nyomtató) felbontásához viszonyított egység, és a *pixel* hosszegység általában az eszköz fizikai képpontjaira vonatkozik. Ha azonban egy megjelenítő eszköz képpontsűrűsége nagyon eltér a tipikus számítógépképernyőjétől, a böngészőnek át kell méreteznie a pixel-értékeket. A *referencia-pixelt* ilyenkor célszerű egy 96 dpi pixelsűrűségű eszköz egy képpontjának a szokásos olvasási távolsághoz (kb. 71 cm) tartozó látószögével definiálni. Így az olvasótávolságra lévő 1 px

1/96 inch-nek, azaz kb. 0,26 mm-nek felel meg. Lézernyomtatón nyomtatva 1 px kb. 0,21 mm, 300 dpi-s nyomtatón kb. 3 képpont (0,25 mm), 600 dpi-s nyomtatón kb. 5 képpont.

Az alábbi kép illusztrálja a látótávolság hatását egy referencia-pixel méretére. A kb. 71 cm-es olvasási távolság 0,26 mm-es pixelt, a 3,5 m-es olvasási távolság 1,3 mm-es pixelt eredményez. (A pixeleknek nagyobbakká kell válniuk, ha nő a látótávolság.)



A következő kép egy készülék felbontásának hatását mutatja be a pixel mértékegységre. Egy *(1px) x (1px)* területet egy alacsony felbontású eszköz (pl. számítógép képernyő) egy pixel-je borít, míg ugyanekkora területet egy nagyfelbontású eszközön (pl. lézernyomtatón) 16 képpont fed le. (Nagyfelbontású eszközön több eszköz-pixel szükséges az egységnyi terület lefedésére, mint alacsony felbontású esetén.)



Számítógép képernyők szokásos felbontása esetében egy hüvelyken kb. 72 képpont van, azaz 1 cm kb. 28,5 képpontnyi távolságot jelent, és a pixel a CSS3-ban már elfogadott lett 1/96 hüvelykes (kb. 0,26 mm) abszolút mértékegységként.

d) funkcionális megnevezéssel (pl. *url*):

Az *URL* (Uniform Resource Locator = egységes forrás meghatározó) a világhálón egy információforrás címét adja meg. Alternatív és általánosabban használt kifejezés rá az *URI* (Uniform Resource Identifier = egységes forrás azonosító), de történeti okokból a kódolásban az *url* használatos. Vesszők, idézőjelek, zárójelek, betű/szóközök, a \ jel (*backslash*) kerülendők.

e) speciális esetek:

e/1. betűcsalád nevek – vesszővel elválasztva betűcsaládnév listaként is megadhatók, kis- és nagybetű érzékenyek, több különálló szó esetén idézőjelek közé kell tenni őket

e/2. színek hexadecimális vagy rgba/hsla definiálása, pl.: #f00; rgba(100, 200, 50, 0.5); stb.

f) egyéb módon: pl. *deg* (fok), *s* (mp), *dpi* (képpont/hüvelyk), stb.

3.5. Az elemi doboz-modell

A CSS stíluslapnyelv egy HTML dokumentum megjelenítésekor az egyes elemekhez láthatatlan dobozokat rendel, melyek magukban foglalhatnak karaktereket, szavakat, sorokat, bekezdéseket, listákat, táblázatokat, képeket, stb. Minden doboznak lehetnek külön formázási tulajdonságai, mint pl. méret, előtérszín, betűtípus, szegély, háttér, stb. A dobozok hierarchikusan egymásba is ágyazódhatnak, pl. egy szó doboza egy lista egy sorának vagy egy táblázat egy cellájának a dobozába, az pedig a lista vagy táblázat dobozába, majd az a weboldal egy szakaszának a dobozába, stb. kerülhet.

A dobozok lehetnek blokkszintű (*block-level box*) dobozok (pl. bekezdés, lista, táblázat, blokkidézet, div szakaszok, stb.), sor- (*line box*) dobozok (pl. egy szöveg egy sora), és soron belüli (*inline-level box*) dobozok (pl. karakterek, szavak, képek egy soron belül).

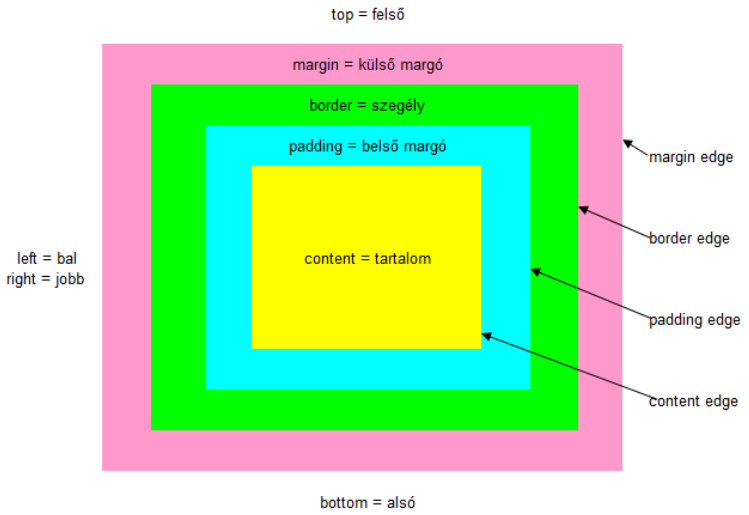
Például az alábbi egyszerű felsorolási lista

```
<ul>
    <li>A lista első eleme.</li>
    <li>A lista második eleme.</li>
</ul>
```

az alábbi CSS dobozokat tartalmazza: egy blokkszintű dobozt az *ul* elemhez, abban két blokkszintű dobozt a két *li* elemnek, azok mindegyike egy-egy sor-dobozt, és mindegyik sor-doboz két-két soron belüli dobozt (egyet a felsorolásjelnek – ami nem is szerepel külön a HTML kódolásban - és egyet a szövegnek).

Megjegyzés: Ha a lista szövege nem férne el egy sorban és ezért sortöréssel több sorban jelenne meg, a *li* elem még több dobozba kerülne.

Egy elemi doboz szerkezetét az alábbi ábra mutatja be:



content = tartalom (szöveg, kép, lista, táblázat, stb.)
content edge = tartalom széle
padding = belső margó, helykitöltés, (szöveg)eltartás
padding edge = belső margó/helykitöltés/(szöveg)eltartás külső széle
border = szegély
border edge = szegély külső széle
margin = (külső) margó
margin edge = (külső) margó külső széle

Egy elemi doboz teljes szélessége tehát:

„content width” + „padding-left” + „padding-right” + „border-left” + „border-right” + „margin-left” + „margin-right”

(tartalom szélessége + bal oldali belső margó + jobb oldali belső margó + bal oldali szegély + jobb oldali szegély + bal oldali külső margó + jobb oldali külső margó)

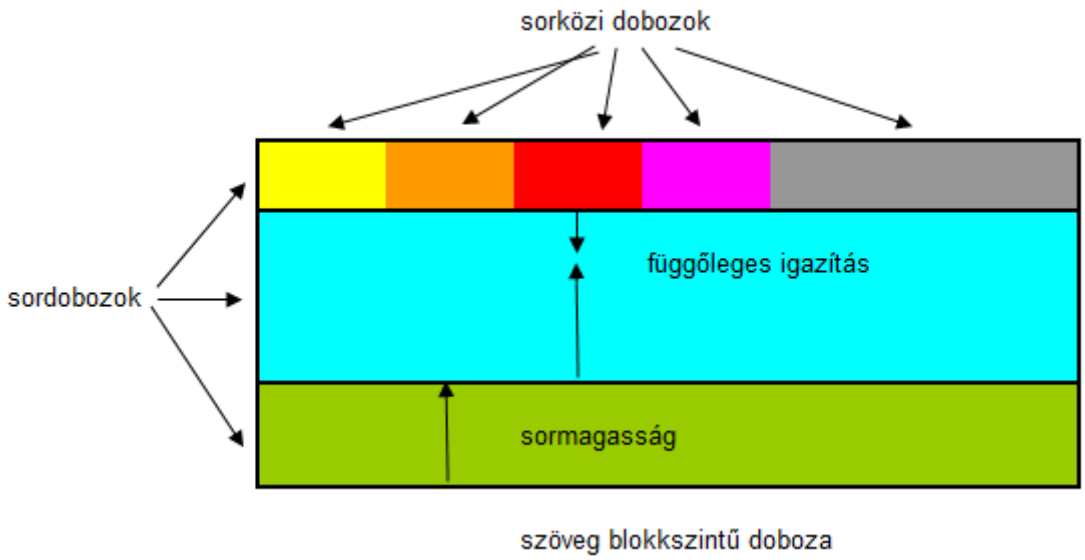
Egy elemi doboz teljes magassága a fentiek analógiájára:

„content height” + „padding-top” + „padding-bottom” + „border-top” + „border-bottom” + „margin-top” + „margin-bottom”

(tartalom magassága + felső belső margó + alsó belső margó + felső szegély + alsó szegély + felső külső margó + alsó külső margó)

A belső margó, szegély és külső margó a doboz egyes oldalain vagy akár mindenhol hiányozhat (valójában nem hiányzik, csak a vastagságának értéke 0). A külső margó vastagsága negatív érték is lehet, ha a szomszédos dobozok részben átfedik egymást - a szegély és belső margó vastagsága nem lehet negatív.

A sor-dobozok (pl. egy szöveg egy sora), és a soron belüli dobozok (pl. karakterek, szavak, képek egy soron belül) is elemi dobozokból épülnek fel.



Az elemi doboz-modell szerkezete és fogalmai összetettebb struktúrákban (táblázat, rugalmas doboz, többhasábos elrendezés, stb.) szintén felhasználhatók.

3.6. A CSS nyelvtana

A CSS stíluslapnyelv a HTML jelölőkódokon belüli elemekhez megjelenítési tulajdonságokat (színt, hátteret, szegélyt, margót, pozícionálást, stb.) rendel. Formázatlan (*plain text*) szöveg, nincsen dokumentumtípus meghatározása, önálló fájl esetén a kiterjesztése *.css.*

3.6.1. A CSS kódolás lehet a HTML dokumentumhoz csatolt önálló fájl(ok)ban külön stíluslap(ok)on (*external style sheet*), más webhelyről letölthetően (*@import rule*), a HTML dokumentum fej részében (*internal style sheet*) vagy egy HTML címkén belül (*inline style*).

a) A szokásos eljárás az egy webhelyen belüli összes HTML oldalhoz **külső stíluslap(ok)** kapcsolása (lásd *A weboldal szerkezete* c. fejezetet):

```
<link rel="stylesheet" href="…….css">
```

b) A külső stíluslap(ok) kapcsolásának a másik módja az **@import** utasítás használata, mely a <*style*>…..<*style*> páros HTML címkével ágyazható be a HTML dokumentum fej részébe:

```
<style>
        @import url(http://www…….)
</style>
```

Ebben az összefoglalóban nem fogjuk használni az *@import* utasítást.

c) Egyoldalas vagy egy különálló weboldal esetén, ill. ha egy webhely egy oldalán a külső stíluslap(ok)hoz képest kis mértékben meg kell változtatni egy HTML lapon a formázást, használatos a **belső vagy beágyazott stílus**, mely a <*style*> páros HTML címkével ágyazható be a HTML dokumentum fej részébe (lásd *A weboldal szerkezete* c. fejezetet):

```
<style>
        CSS formázás kódolása
</style>
```

Ha külső stíluslap is csatlakozik az adott weboldalhoz, akkor a *link* meta-adat **után** kell a *style* meta-adatot kódolni, tehát a beágyazott stílus közelebb legyen a dokumentum törzséhez, mint a külső stílus – hogy a belső stílus felülírja a külsőt.

d) Elvileg lehetséges, de a gyakorlatban csak speciális esetekben ajánlott egy HTML címke *style* jellemzőjének értékeivel definiálni az adott címkére vonatkozó megjelenítést – ezt **szövegközi** (*inline*) stílusnak nevezik. Pl.:

```
<p style="……CSS kódok…">bekezdés szövege</p>
```

Ez a formázás visszatérést jelent a HTML és CSS (tartalom és forma) szétválasztása előtti állapot irányába - ebben az összefoglalóban nem fogjuk használni.

Megjegyzés: A fenti stílusdefiníciók valamennyi médiára (különféle méretű kijelzők, nyomtatás, kivetítés) azonos stílust alkalmaznak.

e) Sajátos a különböző kimeneti eszközökhöz rendelt, egymástól eltérő stílusok meghatározása. A média jellemzők hasonlítanak a CSS nyelvtanához, de van néhány fontos eltérés is:

- a CSS kijelölések és meghatározások arra vonatkozóan tartalmaznak információt, hogy milyen formázással és elrendezéssel jelenjen meg a dokumentum. A *media* jellemző a kimeneti (megjelenítő) eszközre vonatkozó feltétel(ek) leírására használatos.

- a CSS meghatározások szinte végtelen sokféle formázást/elrendezést hozhatnak létre, a *media* jellemző két logikai állapot közül választ – vagy egyezik a megjelenítő eszköz a megadottal, és akkor érvényes a hozzárendelt stílus, vagy nem egyezik, és akkor az a stílus nem érvényesül.

- sok *media* jellemző tartalmazhatja a „*min*-„ és „*max*-„ előtagokat (*prefix*-eket) a „nagyobb vagy egyenlő mint” ill. a „kisebb vagy egyenlő mint” feltételekhez (a HTML-ben a foglalt karakterek „<” ill. ”>” helyett).

- a CSS tulajdonságokhoz mindig értéket is meg kell adni, hogy érvényes meghatározás jöjjön létre. A *media* jellemző használható érték nélkül is, csak a *min*- ill. *max*- előtagosoknál kötelező az érték megadása.

- a *media* jellemzőben felsorolásként egyszerre több média paraméter is definiálható, a felsorolásban a vessző a „logikai vagy”-ot (tehát a több feltétel közül legalább egynek a teljesülését), az *and* a „logikai és”-t (tehát a több feltétel egyidejű teljesülését) jelenti.

- megadható HTML-ként (lásd *HTML Összefoglaló*) és CSS-ként is.

Tehát a különböző médiatípusokhoz és médiafajtákhoz egyedi formázás adható meg, ekkor a ***media*** jellemző értékeivel definiálható az adott média típusa és fajtája. A típusok:

all = *minden* — ez az alapértelmezett, ha nem definiálunk média jellemzőt, vagy az *all* értéket adjuk meg, minden médiában azonos lesz a megjelenítés

screen = *képernyő* — különböző méretű képernyőkre – a képernyő egyéb adottságait is figyelembe vevő – elrendezés és formázás rendelhető ugyanazon dokumentumhoz

print =*nyomtatás* — az *all* vagy *screen* alá jön a kódolásban, és elég csak az azokhoz képesti eltéréseket tartalmaznia

handheld =*kézi eszközök* — a kézi eszköz paramétereihez alkalmazkodó megjelenítés rendelése a dokumentumhoz

projection =*kivetítő* — a kivetítő paramétereihez alkalmazkodó megjelenítés rendelése a dokumentumhoz

Megjegyzés: A *handheld* és *projection* értékek (jelen állás szerint) ki fognak kerülni a szabványból, elegendő tehát a képernyő és nyomtatás értékekkel foglalkozni.

Az egyes média típusokon belül tovább pontosíthatók olyan paraméterek, mint *width*, *height*, *device-width*, *device-height*, *orientation* (portrait, landscape = álló, fekvő), *aspect-ratio* (4/3, 3/2), *device-aspect-ratio* (16/9, 1280/720), *color* (színenkénti bitek száma), *monochrome* (pixelenkénti bitek száma), *resolution* (dpi), *scan* (progressive, interlace).

Egy példa ugyanannak a médiafüggő (400px és 700px közötti szélességű képernyő esetén érvényes) megjelenítési stílusnak a definiálására CSS-ben az *@import* vagy *@media* utasításokkal:

```
@import url(……..css) screen and (min-width:400px) and  (max-width:700px);
vagy
@media screen and (min-width:400px) and (max-width:700px)
{ ……
  …… }
```

3.6.2. A stílusokat kijelölőkkel/kiválasztókkal (*selector*) és meghatározásokkal (*declaration*) lehet megadni.

3.6.2.1. A **kijelölők** vagy kiválasztók (a kettő ugyanazt jelenti, csak fordításbeli a különbség - a továbbiakban a kijelölőt használjuk) azt definiálják, hogy a HTML lap melyik elemére vagy elemeire vonatkozik a meghatározás(ok)ban megadott stílus, azaz a böngészők hogyan jelenítsék meg a kijelölt rész(eke)t. A kijelölő után egy üres betűközzel, kapcsos zárójelek (*curly brackets*) között szerepel(nek) a meghatározás(ok):

kijelölő(k) **{** meghatározás(ok) **}**

Egy kijelölőhöz több meghatározás is tartozhat, ill. több kijelölőre is érvényesíthető(k) ugyanaz(ok) a meghatározások (lásd a továbbiakban).

Megjegyzés: A kijelölés esetenként bonyolultnak tűnhet - amikor az egyes kijelölőket a CSS formázás során használni fogjuk, célszerű ismét visszalapozni ehhez az összefoglalóhoz, és akkor áttekinthetővé és logikussá válik remélhetőleg minden !

A kijelölők lehetnek: elem kijelölők, csoport kijelölők, azonosító kijelölők, származtatott kijelölők, osztály kijelölők, ál-osztály kijelölők, ál-osztály elemek, univerzális kijelölő, negáló kijelölő, jellemző kijelölők és kombinátorok.

a) Az **elem kijelölők** (*type selector*) a weblap összes azonos elemére vonatkoznak, pl. *p* esetén az összes bekezdésre, *q* -nál az összes rövid idézetre, *img* esetén az összes képre, stb.

A dokumentum összes bekezdésére vonatkozó stílus kijelölése a fentiek alapján:

```
p { meghatározás(ok) }
```

b) A **csoport kijelölés** (*group of selectors*) több elemhez ugyanazt a stílust rendeli. A kijelölt elemek vesszővel és üres betűközzel elválasztva sorolandók fel. Pl: a

```
h1, h2 { meghatározás(ok) }
```

kódolás a legnagyobb és második legnagyobb címhez rendel azonos megjelenítést (színt, betűtípust, betűméretet, behúzást, szegélyt, stb.), a *h3*-hoz, *h4*-hez, stb. már nem – azokhoz egy újabb csoport kijelöléssel lehet más megjelenítést hozzárendelni:

```
h1, h2 { meghatározás(ok) }
h3, h4 { más meghatározás(ok) }
```

Figyelem! Ha a csoporton belül az egyik kijelölő érvénytelen, az egész csoport kijelölése érvénytelenné válik, míg ha nem csoportosan, hanem egyenként történik a kijelölés, a többi, nem érvénytelen kijelölés érvényes marad.

c) Az **azonosító kijelölők** (*ID selector*) csak a HTML dokumentumban egyedi *id* jellemzővel ellátott elemre vonatkoznak. Ha pl. három bekezdés van a HTML oldalon, és azok egymástól eltérően formázandók, akkor az egyes bekezdések kezdő címkéihez egyedi *id* jellemzőket kell megadni, így pl. a HTML kódolás:

```
<p id="bek1">.........</p>
<p id="bek2">.........</p>
<p id="bek3">.........</p>
```

CSS-ben a kijelölés a # kettőskereszttel és a jellemzővel történik, a kettő között nincsen betűköz:

```
#bek1 { meghatározás(ok) }
#bek2 { más meghatározás(ok) }
#bek3 { megint más meghatározás(ok) }
```

d) A **származtatott kijelölők** (*descendant selector*) - néha **környezetfüggő kijelölők**nek is fordítják őket – egy *id* jellemzővel definiált szakaszon belüli elemkijelölést végeznek. Ha a HTML oldal például két szakaszból áll, és az egyes szakaszokban lévő bekezdésekhez szakaszonként eltérő megjelenítést tervezünk, akkor a HTML oldal kódolása:

```
<div id="szak1">
      <p>.................................</p>
      <p>.................................</p>
</div>
<div id="szak2">
      <p>.................................</p>
      <p>.................................</p>
</div>
```

CSS-ben az egyes szakaszok bekezdéseihez a megjelenítések kódolása:

```
#szak1 p { meghatározás(ok) }
#szak2 p { más meghatározás(ok) }
```

Az azonosító kijelölőt tehát egy elemkijelölő követ, köztük egy üres betűköz van.

Figyelem! A csoport kijelölőt és a származtatott kijelölőt formailag csak egy vessző választja el egymástól, de teljesen más elemeket fognak kijelölni. A *cite, abbr* (van közöttük vessző) az összes címhivatkozást és összes rövidítést kijelöli, míg a *cite abbr* (nincsen közöttük vessző) csak a címhivatkozásokban lévő rövidítéseket jelöli ki.

e) Az **osztály kijelölők** (*class selector*) általunk osztályokba sorolt elemek azonos megjelenítését határozzák meg. Például ha a HTML oldal bekezdéseinek egy csoportjához egy stílust, egy másik csoportjához egy másik stílust rendelünk, akkor a HTML oldalon a kódolás:

```
<p class="stilus1">.........................</p>
<p class="stilus2">.........................</p>
<p class="stilus1">.........................</p>
<p class="stilus2">.........................</p>
```

A CSS kódolás a megjelenítésre:

```
p.stilus1 { meghatározás(ok) }
p.stilus2 { más meghatározás(ok) }
```

A címke és az osztálynévhez tartozó pont között nincsen üres betűköz.

Nem csak azonos, hanem különböző címkékhez (pl. *h1, p, blockquote*, stb.) is rendelhetők közös osztályok, ekkor a CSS kódolás a címkéket nem tartalmazza:

```
.stilus1 { meghatározás(ok) }
.stilus2 { más meghatározás(ok) }
```

Az osztálykijelölők ponttal kezdődnek, a pont és az osztálynév között nincsen üres betűköz. A HTML lapon természetesen az egyes címkéknél be kell írni a *class* jellemzőt, különben nem lesz a stílusmegadásnak hatása.

A weboldal egy része stílusának kijelöléséhez a kijelölők, azonosítók és osztályok kombinációja is használható. Pl.: az alábbi CSS kódolás

```
#header p.stilus1 { meghatározás(ok) }
```

csak a weblap *fejléc* div-szakaszában lévő bekezdések közül a *stilus1* osztályba tartozókhoz rendeli a meghatározás(ok)ban definiált megjelenítést.

f) Az **ál-osztály** (látszólagos osztály, pszeudo-osztály) **kijelölők** (*pseudo-class selector*) öt alcsoportba oszthatók:

f/1. *link*-típusú ál-osztályok:

```
a:link { meghatározás(ok) }
```
(link = még nem meglátogatott hivatkozás)

```
a:visited { meghatározás(ok) }
```
(visited = felkeresve, látogatva a hivatkozás)

f/2. dinamikus ál-osztályok:

```
:hover { meghatározás(ok) }
```
(*hover* = egy elem által létrehozott dobozra mutatás egérrel - a kurzor az elemdoboz felett van, de nincsen kattintás vagy az egérgomb lenyomása)

```
:active { meghatározás(ok) }
```
(*active* = egy elem által létrehozott doboz aktiválása egérrel - a kurzor az elemdoboz felett van lenyomott bal egérgombbal, és nem engedjük fel a bal egérgombot)

```
:focus { meghatározás(ok) }
```
(*focus* = egy elem által létrehozott doboz fókuszba kerül egy egérkurzor vagy billentyűzet esemény következtében)

Természetesen származtatott kijelöléssel a fenti ál-osztály kijelölések pontosíthatók, szűkíthetők, pl.:

```
#nav a:visited { meghatározás(ok) }
#container:hover { meghatározás(ok) }
```

Az ***a*** horgonyhoz több, egymást nem kizáró link- és dinamikus ál-osztályú kijelölés is kombinálható, pl. ha a meglátogatott hivatkozásra ismét egérkurzorral rámutatunk. Ekkor a kijelölés:

```
a:visited a:hover
```

Ha az ***a*** horgonyhoz több link- ill. dinamikus álosztály tartozik, kötött a sorrend:

```
a:link a:visited a:hover a:active
```

f/3. target-típusú ál-osztály:

Hivatkozásoknál az ugrási célpont megjelenítéséhez a **:target** kiválasztót lehet használni.

f/4. checked-típusú ál-osztály:

Űrlapoknál a bejelölt választógomb vagy jelölőnégyzet formázásához a **:checked** kiválasztót lehet használni.

Az f/1.-f/4. azért ál-osztályok, mert állapotuk nem a HTML kódtól, hanem a weboldal látogatójának tevékenységétől függ.

f/5. lang-típusú ál-osztály:

A nyelv szerinti kiválasztás a **:lang** (lang= *language* = nyelv **)** kijelölővel végezhető el.

g) A pszeudo-elemek két csoportba sorolhatók:

g/1. Blokkszintű elemek első formázott soraira és/vagy első betűire definiálható külön stílus a **::first-line** és **::first-letter** kijelölőkkel.

Pl. ha minden bekezdés első sorát jelöljük ki:

```
p::first-line { meghatározás(ok) }
```

Minden bekezdés első betűje esetén (pl. iniciálék kialakításakor):

```
p::first-letter { meghatározás(ok) }
```

Az első sor és első betű azért ál-osztályok, mert állapotuk a dokumentumban betöltött helyüktől függ - az első sor tartalma pl. egy betű- vagy szakasz méretváltozás hatására megváltozhat.

g/2. A forrásdokumentumban nem szereplő tartalmat lehet beilleszteni a **::before** és az **::after** kijelölőkkel. Pl. ha minden bekezdés elé ugyanazt az elemet illesztjük be:

```
p::before { meghatározás(ok) }
```

Ha utánuk:

```
p::after { meghatározás(ok) }
```

Figyelem! A pszeudo-osztályok és pszeudo-elemek megkülönböztetését szolgálja az előttük lévő egy, ill. kettő kettőspont.

h) Kevéssé használt kijelölő a * (*universal selector*), mely a dokumentumban lévő minden egyes elemet kijelöl, és a *:not( )* negáló (*negation*) kijelölő, mely a zárójelben lévő elem kijelölését tiltja le. Kettejük kombinációja a **:not( )*, ami a zárójelben megadott elem kivételével mindent kijelöl. Az univerzális kijelölőt használják az osztály kijelölőkkel is, melynek nincsen külön jelentése, tehát a **.stilus1* és a *.stilus1* egyaránt valamennyi *stilus1* osztályba sorolt elemre vonatkozó kijelölést jelentik.

i) A **jellemző kijelölő** (*attribute selector*) vagy csak a jellemző alapján, vagy a jellemző és annak meghatározott értéke alapján jelöl ki.

A *h1[title]* a *title* jellemzővel és tetszőleges értékkel rendelkező *h1* címsorokat, a *h1[title="pelda"]* a *title* jellemzőjű és *pelda* értékű *h1* címsorokat, a *h1[title~="pelda"]* a *title* jellemző szóközzel elválasztott, felsorolt értékeinek listájában a *pelda* értéket is tartalmazó *h1* címsorokat jelöli ki.

A *h1[title*="eld"]* az elsődleges címsorok közül azokat jelöli ki, melyek *title* jellemzője olyan értékkel rendelkezik, mely valamilyen formában tartalmazza az *eld*-et, míg a *h1[title^="pel"]* a *pel*-el kezdődő, a *h1[title$="da"]* a *da*-val végződő értékűeket jelöli ki.

Megjegyzés: Végül az utolsó két kijelölő fajtához - az un. strukturális pszeudo-osztályokhoz és a kombinátorokhoz – a következő fejezetben az öröklésre vonatkozó rész el-olvasása is szükséges.

f/5. strukturális ál-osztályok:

A strukturális ál-osztályok lehetővé teszik a dokumentumfa által tartalmazott strukturális információk felhasználásával a más kijelölőkkel egyszerű módon nem elérhető kiválasztást. Fajtái:

- *:root* pszeudo-osztály a dokumentum gyökér-elemét (*html*) képviseli

- *:nth-child()* pszeudo-osztály általános képlete *:nth-child(an+b)*, mely olyan elemet képvisel, amelynek van egy szülő eleme és a dokumentumfában *an+b-1* testvére. A megkötések:

  - az *n* nulla vagy pozitív egész szám
  - *a* és *b* egyaránt lehet nulla vagy pozitív/negatív egész szám
  - *an+b* összege nulla vagy pozitív egész szám

Ha *a* és *b* is nulla, akkor az *:nth-child(0)* ál-osztály semmilyen elemet nem jelöl ki, hiszen az első gyermek indexe 1, nullás gyermek nem létezik.

Ha *an+b* összege nulla vagy negatív, ismét a fenti eset áll elő – semmilyen elemet nem jelöl ki, mert negatív vagy 0 indexű gyermek nem létezik, hiszen 1-el indul az indexálásuk.

Ha *a* nulla és *b* nem az, akkor *:nth-child(b)* a *b*-edik gyermeket jelöli ki. Pl. az *:nth-child(3)* egy szülő elem harmadik gyermekét képviseli.

Ha *b* nulla és *a* nem az, akkor *:nth-child(an)* minden *a*-adik gyermeket jelöli ki. Pl.: a=1 esetén (*:nth-child(n))* az 1, 2, 3,…..indexű gyermekeket, vagyis mindegyiket, a=2 esetén (*:nth-child(2n))* a 2, 4, 6,…….vagyis a páros indexű gyermekeket, stb.

Ha sem *a* sem *b* nem nulla, és *an+b* összege pozitív szám, akkor pl. *:nth-child(10n-1)* a 9-edik, 19-edik, 29-edik,……gyermeket jelöli ki, és ugyanezt a kijelölést valósítja meg *:nth-child(10n+9)* is. Az *a* és *b* önmagában tehát lehet negatív, de csak az *an+b* összeg pozitív értékei esetén értelmezett - így érvényes pl. a fenti *:nth-child(10n-1)* ha *n>0*, vagy az *:nth-child(-n+6)* (mely az első hat gyermeket jelöli ki) ha n<7.

Az *:nth-child(2n+1)* felveheti az *odd* (páratlan), az *:nth-child(2n)* az *even* (páros) argumentumot is.

Megjegyzés: A kijelölőben megengedett az üres betüköz a nyitó zárójel után és a záró zárójel előtt, továbbá a „+" és „-„ jel mindkét oldalán (mely elválasztja egymástól az *an* és *b* részeket). Így pl. érvényesek: *:nth-child(3n + 1), :nth-child(-n+ 6),* érvénytelen viszont pl: *:nth-child(3 n).*

- *:first-child* pszeudo-osztály olyan elemet képvisel, mely valamilyen másik elem első gyermeke, megegyezik az *:nth-child(1)*-el. Pl.: *div > p:first-child* kijelölő a div elem első *p* elemét képviseli.

- *:nth-last-child()* pszeudo-osztály általános képlete *:nth-child(an+b),* mely olyan elemet képvisel, amelynek van egy szülő eleme és a dokumentumfában **utána** van *an+b-1* testvére. Szintaxisa azonos a *:nth-child()* –éval, és szintén érvényesek az *odd* és *even* argumentumok. Pl.: *tr:nth-last-child(-n+2)* egy táblázat utolsó két sorát jelöli ki (az utolsótól visszafelé számolva).

- *:last-child* pszeudo-osztály valamilyen szülő elem utolsó gyermekét képviseli, megegyezik a *:nth-last-child(1)* –el. Pl.: egy *ol* lista utolsó *li* listaelemét képviseli az *ol > li:last-child.*

- *:only-child* pszeudo-osztály megegyezik a *:first-child, :last-child, :nth-child(1)* és *:nth-last-child(1)* kijelölésekkel.

- *:nth-of-type()* pszeudo-osztály általános képlete *:nth-of-type(an+b),* mely olyan elemet képvisel, melynek van egy szülője és **előtte** ugyanavval az elemnévvel rendelkező *an+b-1* testvére. Szintaxisa azonos a *:nth-child()* –éval, és szintén érvényesek az *odd* és *even* argumentumok. Pl. felváltva változtatott tulajdonságú képeket az *img:nth-of-type(2n+1)* és az *img:nth-of-type(2n)* -al lehet kijelölni.

- *:first-of-type* pszeudo-osztály megegyezik az *:nth-of-type(1)*-el, vagyis egy szülő gyermekei között az azonos elemnévvel rendelkező testvérek közül az elsőt jelöli ki. Az előző példa esetén *img:first-of-type* (a sokfajta testvér elem közül) az első *képet* jelöli ki.

- *nth-last-of-type()* pszeudo-osztály általános képlete *:nth-last-of-type(an+b*) olyan elemet képvisel, melynek a dokumentumfában egy szülője és **utána** *an+b-1* testvére van ugyanavval az elemnévvel. Szintaxisa azonos az *:nth-child()*–éval, és az *even* és *odd* értékeket is elfogadja. Pl.: *egy body* valamennyi *h2* gyermeke – kivéve az elsőt és az utolsót – kijelöléséhez az alábbi használható:

```
body > h2:nth-of-type(n+2):nth-last-of-type(n+2)
```

- *:last-of-type* pszeudo-osztály megegyezik az *:nth-last-of-type(1)* –el. Pl. egy táblázat utolsó celláját kijelöli a *tr > td:last-of-type,* vagy az előző *h2* elemeket kiválaszthatjuk az alábbi módon is:

```
body > h2:not(:first-of-type):not(:last-of-type)
```

- *:only-of-type* pszeudo-osztály megegyezik a *:first-of-type, :last-of-type, :nth-of-type(1)* és *:nth-last-of-type(1)* kijelölésekkel.

i) A **kombinátorok** (*combinator*) az öröklési viszonyok – tehát a családfa mintáját követő dokumentumfa - alapján történő kijelölést tesznek lehetővé.

i/1. származtatott kombinátor

Egy elem - mint egy másik elem leszármazottja - jelölhető ilyen módon ki. Az első kijelölés az ős, a második kijelölő a leszármazott. A származtatott kombinátor egy üres betűköz (*whitespace*), mely a két kijelölés között helyezkedik el. Pl. ha egy formázásra kijelölt címrészben van egy tartalmilag kiemelt elem, azaz

```
<h1>Ez a<span class="fontos">cím<em>nagyon-nagyon</em>
    fontos.</span></h1>
```

akkor annak kijelölése származtatott kombinátorral:

```
h1 em
```

A ***div * p*** olyan bekezdéseket jelöl ki, melyek a *div* unokái vagy még további leszármazottai. Az univerzális kijelölő ("*") két oldalán lévő üres betűköz nem az univerzális kijelölőhöz, hanem a kombinátorhoz tartozik. Azt jelöli, hogy a *div* valamilyen elemnek az őse, és ez a valamilyen elem a *p* őse.

Származtatott kombinátor alkalmazható összetettebb kijelölésekben is, pl.: ***div p *[****href****]***

amely a *href* jellemzőjű elemeket jelöli ki, melyek egy *p* bekezdésben vannak, amely pedig egy *div* tárolóban van.

i/2. gyermek kombinátor

Két elem közötti szülő-gyermek viszonyt ír le. Jelölése a „nagyobb mint” (>) jel, az első kijelölő a szülő, a második a gyermek elem (a betűköz opcionális). Pl. ha a *p* elem a *body* elem gyermeke, akkor:

```
body > p
```

A származtatott és gyermek kombinátorokat vegyesen alkalmazva a

```
div ol>li p
```

egy olyan *p* elem kijelölése, mely egy *li* elem leszármazottja, amely egy *ol* elem gyermeke, ami viszont egy *div* leszármazottja.

i/3. testvér kombinátorok

Két elem közötti testvéri viszonyt (közös szülő, de egymásnak nem leszármazottai) ír le. Két fajtájuk a „szomszédos testvér” és az „általános testvér” kombinátor.

- a „szomszédos testvér” kombinátor a két kijelölő közé helyezett plusz jellel van jelölve (+). Ezeknek közös a szülőjük a dokumentfában, és az első elem a sorban közvetlenül megelőzi a a másodikat. Pl: *h1* elemet közvetlenül követő *p* elem kijelölése:

```
h1 + p
```

Ha pl. a *h1* elemre egy olyan korlátozást is adunk, hogy a „*piros*” osztályba kell tartoznia, akkor a *p* kijelölése:

```
h1.piros + p
```

- az „általános testvér” kombinátor a két kijelölő közé helyezett „*tilde*” jellel (~) van jelölve. Ezeknek közös a szülőjük a dokumentumfában, és az első elem a sorban (nem feltétlenül közvetlenül) megelőzi a másodikat. Pl: egy *h1*-et valamikor követő *pre* elem kijelölése:

```
h1 ~ pre
```

3.6.2.2. A **meghatározás** két részből áll, a tulajdonságból (*property*) és az értékből (*value*).

A **tulajdonság** (angolul) azt definiálja, hogy a kijelölt rész milyen paraméterét (tulajdonságát), az **érték** pedig, hogy azt milyenre akarjuk változtatni. A tulajdonság után betűköz nélkül kettőspontot, az érték elé egy üres betűközt, utána üres betűköz nélkül pontos-

vesszőt kell rakni. Az érték csak akkor teendő idézőjelbe, ha két különálló szóból áll (a kötőjeles szó is egy szónak számít), egyébként nem szabad idézőjelet használni:

```
h1, h2 { background-color: green; }
h1, h2 { font-family: "Times New Roman"; }
```

Az első kód a *h1* és *h2* címsorokhoz zöld háttérszínt, a második kód a *h1* és *h2* címsorokhoz *Times New Roman* betűcsaládot rendel (lásd következő fejezeteket).

Egy kijelölőhöz több tulajdonság/érték páros is rendelhető - ekkor egy üres betűközt kell a pontosvesszők után hagyni. A fenti két kód összevontan (*shorthand* = összevont, rövid alak):

```
h1, h2 { background-color: green; font-family: "Times New
         Roman"; }
```

Ha nem vagyunk benne biztosak, hogy az általunk megadott értéket minden böngésző értelmezi, egyes tulajdonságoknál vesszővel és üres betűközzel elválasztva több értéket is megadhatunk a preferenciasorrendünk szerint – ekkor az adott böngésző először az első értéket választja, ha azt nem tudja értelmezni, akkor a másodikat, stb. Ez az eset a *Betűtípusok* fejezetben fordul elő, például:

```
h1, h2 { background-color: green; font-family: Arial, "Times
         New Roman" sans-serif; }
```

Az áttekinthetőség érdekében a stílust célszerű könnyebben olvashatóan, tagoltan kódolni az alábbi szerint:

```
h1, h2 {
         background-color: green;
         font-family: Arial, sans-serif, "Times New Roman"; }
```

A tagolt, egymás alá írt kódolás és a különböző összevont alakok használatának mellőzése – a jobb áttekinthetőség áraként – nagyobb fájlméretet eredményez.

4.6.3. A CSS kódolásba is beírhatók **megjegyzések**, melyek a megjelenítésre nincsenek hatással, de a későbbiekre nézve emlékeztetők, észrevételek tehetők a stíluslapon. Akár egy teljes stíluslap vagy annak egy része is megjegyzésbe foglalható, ilyen módon ideiglenesen hatályon kívül helyezve, de mégis megőrizve a megjegyzésbe rakott CSS kódolást:

```
/* ……megjegyzés…… */
```

3.7. A CSS hierarchiája

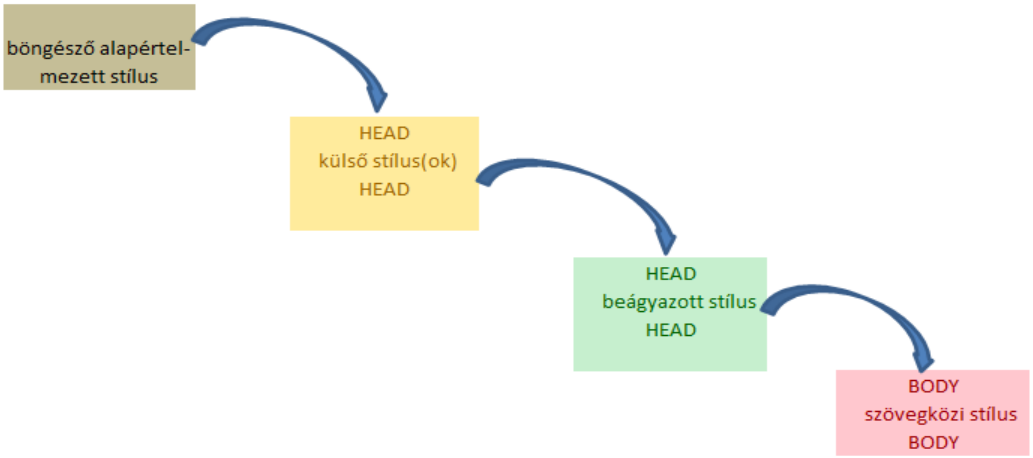
A *Cascading Style Sheets* nevében a „Cascading” szó szokásos magyar fordításai „lépcsőzetes”, vagy „rangsorolt”, vagy „egymásba ágyazott”, vagy „lépcsőzetesen egymásra épülő” – ezek mind utalnak a stílusok rangsorolt egymásra rétegződésére illetve egymásba ágyazódására, és a stílusok közötti esetleges ütközések feloldásának módjaira.

Egy weboldal elemeire ugyanis általában több stílusmeghatározás egyidejűleg vonatkozik – ekkor a tényleges megjelenítést a rangsor, öröklés és szűkítés szabályai határozzák meg.

3.7.1. **A rangsor** azt jelenti, hogy minél közelebb van egy stílusmegadás a formázandó elemhez, annál nagyobb a hatása az adott elem stílusára, (tehát annál feljebb van a rangsorban), azaz felülírja az elemtől távolabbi (tehát a rangsorban alatta lévő) stílusokat.

Egy weboldal egy elemére vonatkozó stílusmeghatározás – az előző fejezetben leírtak szerint – lehet:

- az elem HTML címkéje jellemzőjeként szövegközi (*inline*) stílus formájában,
- és/vagy beágyazott/belső (internal) stílusként a HTML dokumentum fej részében a meta-adatok között,
- és/vagy külső (external) stílus csatolása ugyancsak a HTML dokumentum fej részében a meta-adatok között,
- és/vagy a böngésző alapértelmezett megjelenítési tulajdonságaiban.



A fenti felsorolás és ábra egyben a rangsort is leírja, hiszen:

- a szövegközi stílusnál már nem lehet közelebb menni egy elemhez, mivel a címke jellemzőjeként a nyitó címkébe van kódolva a stílus
- a belső és külső stílus megadásánál előírtuk, hogy a HTML oldal fej részében a belső stílus a külső stílus után következzen, azaz közelebb legyen a belső stílus a HTML oldal törzs részébe kódolt elemekhez mint a külső stílus
- a böngészők alapértelmezett beállításai csak ott érvényesülnek, amire a weblap készítője nem adott meg semmilyen stílust.

Megjegyzés: A böngészőkben a felhasználó beállíthatja saját/felhasználói stílusát, ami felülírja a weblap valamennyi stílusutasítását. Ezt a webszerkesztő nem tudja - de nem is feladata - befolyásolni.

3.7.2. Az **öröklés** azt jelenti, hogy ha egy elem az elrendezési struktúrában őt megelőző másik elembe ágyazódik bele, akkor a megelőző elem a hierarchiában felette áll, és - ha arra nincsen megadva külön stílus - örökíti (*inherit*) a stílusát a hierarchiában alatta álló elemre. Az elemek folyamatosan csökkenő hierarchiája az egész dokumentumot kijelölő HTML elemmel kezdődik, a *body* elemmel folytatódik, a törzsben lévő, egymást követő blokkszintű elemek a következő lépcső, de ha pl. egy *div*-be címsor és bekezdések vannak beágyazva, akkor az már ismét egy öröklési szint, stb.

Behúzásokkal jelölve a beágyazásokat a fenti egyszerű hierarchia sematikus kódolása:

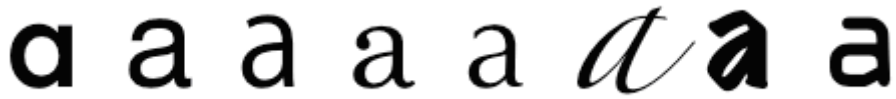
```
<html>
    <head>……..</head>
    <body>
        <div>
            <h1>……</h1>
            <p>……..</p>
            <p>……..</p>
        </div>
        <div>……</div>
    </body>
</html>
```

A *body* örökíti tulajdonságait a teljes tartalomnak, szülő-gyermek (szülő=*parent*, gyermek=*child*) viszonyban van mindkét *div* elemmel. A két *div* testvérek (testvér=*sibling*), tulajdonságokat egymástól nem, csak a *body* szülőjüktől örökölnek. A *h1*-nek és a két *p*–nek az első *div* a szülője, tőle örökölnek, a második *div*-től nem. A *h1* és a két *p* testvérek, egymástól nem örökölnek tulajdonságokat, csak az első *div* szülőjüktől.

3.7.3. A **szűkítés** azt jelenti, hogy a szűkebb (pontosabb) kijelölők felülbírálják az általánosabb kijelölőket. Például mind az osztálykijelölőknek, mind az azonosítókijelölőknek nagyobb a súlyuk mint az elemkijelölőknek, tehát felülírják azok tulajdonságait.

3.8. Betűtípusok

Egy karakternek nagyon sok lehetséges megjelenítési változata van, pl.:



Egy betű-család (*font-family*) általában egy alapként szolgáló betűtípusból és annak pár változatából (félkövér, dőlt, és még lehet több fajta is) áll.

A betűtípus (*font*) olyan karakterek – betűk, számok, írásjelek – együttese, amelyek bizonyos közös jellemzőkkel rendelkeznek. A betűtípus forrása lehet helyi (abban a rendszerben, amelyben a böngésző fut) vagy letölthető (*web fonts*). A betűtípus tervezők sok munkát fektetnek egy-egy *font* kifejlesztésére, ezért a letölthető betűtípusok általában licencedíjasok.

A CSS-ben az egyes betű-képek (*font face*) definiálása a betűtípus-családból és egyéb betű-tulajdonságok felsorolásából állnak, ez lehetővé teszi a beállítások egymástól független változtatását. Például:

A betű-család (pl. a fentiekben a Futura) csak a betű-képek egy készletét tartalmazó nevet specifikál, az egyedi betű-kép további kiegészítésekkel pontosítható.

A betű-családnevek két fő típusa:

a) **általános** vagy **gyűjtő** családnevekből (*generic family keywords*) öt fajta van:

serif (betűlábat lezáró talpas), *sans-serif* (talp nélküli), *monospace* (azonos szélességű), *cursive* (kurzív), *fantasy* (fantázia).

A legismertebb és leggyakrabban alkalmazottak a **serif** és **sans-serif**:

A **monospace** esetében az egyetlen kritérium, hogy valamennyi karakterjelnek azonos, rögzített szélessége legyen. Ezt általában számítógépkód minták megjelenítésénél használják:

A **kurzív** (dőlt betűs, folyóírásszerű) a karakterjelek részbeni vagy teljes összekötésével inkább kézzel írt mint nyomtatott írásképhez hasonlít:

A **fantázia** elsősorban játékos, dekoratív betűtípusokat használ:

A böngészők gyakran nem ismerik fel a kurzív és fantázia általános betű-családneveket.

b) A **specifikus** családnevekből (*font family name*) nagyon sok van, és nem mindegyiket ismerik fel a felhasználók gépén a böngészők. Általában prioritási sorrendben felsorolva, a többi CSS-tulajdonságtól eltérően vesszővel elválasztva adandók meg (ezzel jelezve, hogy alternatívákról van szó), utolsóként egy általános családnévvel lezárva a sort. A

böngésző addig halad a specifikus családnevek listájában, amíg nem talál egy számára rendelkezésre álló betű-típust, a gyűjtő családnév pedig általános tartalékként (*general fallback*) szolgál arra az esetre, ha a felsorolt specifikus családokból egy sem lenne megjeleníthető.

Egy megbízhatóan web-biztos betű-családnév gyűjtemény a tartalék betű-családdal párosítva:

font-family: Arial, Helvetica, sans-serif;
font-family: 'Arial Black', Gadget, sans-serif;
font-family: 'Bookman Old Style', serif;
font-family: 'Comic Sans MS', cursive;
font-family: Courier, monospace;
font-family: 'Courier New', Courier, monospace;
font-family: Garamond, serif;
font-family: Georgia, serif;
font-family: Impact, Charcoal, sans-serif;
font-family: 'Lucida Console', Monaco, monospace;
font-family: 'Lucida Sans Unicode', 'Lucida Grande', sans-serif;
font-family: 'MS Sans Serif', Geneva, sans-serif;
font-family: 'MS Serif', 'New York', sans-serif;
font-family: 'Palatino Linotype', 'Book Antiqua', Palatino, serif;
font-family: Symbol, sans-serif;
font-family: Tahoma, Geneva, sans-serif;
font-family: 'Times New Roman', Times, serif;
font-family: 'Trebuchet MS', Helvetica, sans-serif;
font-family: Verdana, Geneva, sans-serif;
font-family: Webdings, sans-serif;
font-family: Wingdings, 'Zapf Dingbats', sans-serif;

A betű-családok neveit csak akkor szabad idézőjelbe tenni, ha több szóból állnak (a kötőjeles szavak egy szónak számítanak) – akkor viszont kötelező az idézőjel használata. A családnevek kisbetű/nagybetű érzékenyek - a specifikus családnevek általában nagybetűvel kezdődnek. Példa egy HTML lap törzsének betűcsalád-megadására a kijelölő - tulajdonság - érték CSS kódolást használva:

```
body { font-family: „Arial Black”, Gadget, sans-serif; }
```

A **font-weight** (betű-vastagság) tulajdonság a betű-típusban a karakterjel súlyát vagy a vonalvastagságot specifikálja. Az értéke 100 és 900 közötti számmal vagy névvel adható meg.

A számokkal történő megadás:

100 - hajszálvékony (*thin*)
200 - extra vékony (*extra light*)
300 - vékony (*light*)
400 - normál (*normal*)
500 - közepes (*medium*)
600 - enyhén félkövér (*semi bold, demi bold*)
700 - félkövér (*bold*)
800 - extra félkövér (*extra bold, ultra bold*)
900 - fekete, vastag (*black, heavy*)

Névvel történő megadás:

normal: megfelel a 400-nak
bold: megfelel a 700-nak
bolder: az öröklött értéknél kövérebb
lighter: az öröklött értéknél vékonyabb

Gyakran csak néhány vastagság szerepel egy betű-családban, ilyenkor a böngésző egy hozzá közeli vastagságot választ (feketével a rendelkezésre álló vastagságok):

A *bolder* és *lighter* értelmezése (az öröklött vastagság-értékhez képest):

örökölt vastagság értéke	**bolder**	**lighter**
100	400	100
200	400	100
300	400	100
400	700	100
500	700	100
600	900	400
700	900	400
800	900	700
900	900	700

Aki a vastagság értékeket pontosabban akarja szabályozni, inkább a numerikus meghatározást használja, mint a relatív (*bolder* és *lighter*) vastagságokat.

A **font-stretch** (betű-kiterjedés) tulajdonság *normal* (normál), *condensed* (összenyomott) és *expanded* (megnyújtott) betű-képet választ egy betű-családból. A megnevezéseik *(keywords*):

- ultra condensed (ultra-összenyomott)
- extra condensed (extra-összenyomott)
- condensed (összenyomott)
- semi condensed (félig összenyomott)
- normal (normál)
- semi expanded (félig megnyújtott)
- expanded (megnyújtott)
- extra expanded (extra-megnyújtott)
- ultra expanded (ultra-megnyújtott)

Ha egy betű-kép nem létezik egy adott kiterjedésre, a *normal* és *condensed* a keskenyebb kiterjedés felé, az *expanded* a szélesebb irányába keres helyettesítő értéket.

Az alábbi ábra azt mutatja, hogy a kilenc betű-kiterjedés tulajdonság hogyan befolyásolja a betű-választást, ha a betű- családban több szélesség van (a szürke azt a szélességet jelöli, amire nincsen betű-kép), és ezért más szélesség helyettesítődik:

Az örökölt értékhez viszonyítva a **wider** (szélesebb) és **narrower** (keskenyebb) relatív értékmegadás is használható, ekkor az alábbi táblázat szerint történik a megjelenítés:

örökölt érték	**wider**	**narrower**
ultra-condensed	normal	condensed
extra-condensed	normal	condensed
condensed	normal	condensed
semi-condensed	normal	condensed
normal	expanded	condensed
semi-expanded	expanded	normal
expanded	expanded	normal
extra-expanded	expanded	normal
ultra-expanded	expanded	normal

Figyelem! Az örökölt értékhez viszonyított *wider* és *narrower* értékek kevéssé egzaktak és elterjedtek, akár törlésre is kerülhetnek a szabvány következő verzióiból.

A **font-style** (betű-stílus) tulajdonság értéke lehet *normal* (normál), *italic* (dőlt) vagy *oblique* (ferde). Az *italic* általában *cursive*, míg az *oblique* tipikusan a *normal* betűképnek egy megdöntött verziója. (Ha az *italic* nem érhető el, a böngésző *oblique*-ot választ, ha az sincs, akkor egy *oblique* szintetizálható egy *normal* betűkép döntési transzformációjával.)

A **font-variant** (betű-változat) tulajdonság értéke hagyományosan *normal* (normál) vagy *small-caps* (kiskapitális – mely a kisbetűket olyan nagybetűkké alakítja, melyeknek mérete kisebb mint a tényleges nagybetűké). A jelenlegi fejlesztések a *font-variant-caps, font-variant-ligatures, font-variant-position, font-variant-numeric* és *font-variant-alternates* tulajdonságokra irányulnak (ezeket még nem támogatják a böngészők), melyek összevont alakjává válik a *font-variant*.

A **font-size** (betűméret) tulajdonság értéke definiálható *absolute-size* (abszolút méretben) vagy *relative-size* (relatív méretben).

Az abszolút méret a következőképpen adható meg megnevezéssel:

xx-small, x-small, small, medium, large, x-large, xx-large.

A *medium* értéket referenciának véve a böngészők az alábbiak szerint alakítják a betűméreteket:

absolute-size értékek	**xx-small**	**x-small**	**small**	**medium**	**large**	**x-large**	**xx-large**	
méretező faktor	3/5	3/4	8/9	1	6/5	3/2	2/1	3/1
HTML címsorok	h6		h5	h4	h3	h2	h1	
HTML betű méretek	1		2	3	4	5	6	7

Az abszolút méret ugyancsak megadható egy abszolút számmal vagy százalékosan (ez az öröklött betűmérethez képesti százalékos értéket jelenti). A százalékos érték vagy *em* érték látványos, lépcsőzetes stílust eredményez.

A **relative-size** (relatív méret) az öröklött betűmérethez képest növeli vagy csökkenti a méretet, lehetséges értékei *larger* (nagyobb) és *smaller* (kisebb). Pl. ha az öröklött betűméret *medium*, akkor a *larger* érték hatására az elem betűmérete *large* lesz, stb.

Egy szöveg olvashatóságát bármely betűméret esetén a betűtípus és a kisbetűk és nagybetűk aránya (*aspect ratio*) is befolyásolja. Tartalék betűtípussal (*fallback font*) történő megjelenítés esetén nem feltétlenül lesz ugyanaz a tényleges betűméret és a kisbetűk és nagybetűk aránya mint az eredetileg tervezett betűtípusnál, és ez ronthat az olvashatóságon. A **font-size-adjust** (betűméret igazítás) tulajdonság ilyen esetben ad módszert a betűképek átméretezésére a szöveg eredeti olvashatóságának megőrzése céljából – ennek a számításnak a részleteibe nem megyünk bele, mert a böngészők még nem értelmezik ezt a tulajdonságot.

A **font** tulajdonsággal összevontan (*shorthand*) is megadhatók az egyes betű-tulajdonságok, a felsorolás sorrendje kötött: *font-style, font-variant, font-weight, font-size, font-family*.

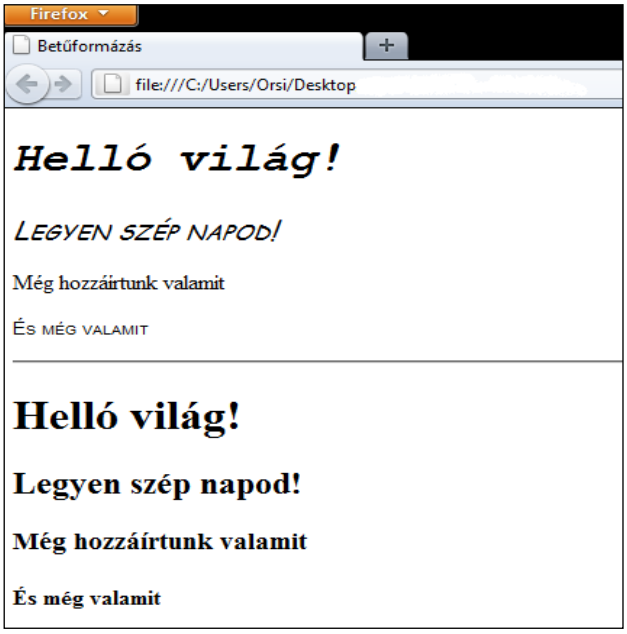
A *font-stretch* és *font-size-adjust* tulajdonságok nem vonhatóak be az összevont alakba - ez a korábbi CSS verziók irányába való visszafelé kompatibilitás fenntartása miatt van így - ezeket külön kell definiálni.

A négy címsoros, kezdő HTML lapunkra alkalmazva a betűformázási lehetőségeket:

```
h1 { font-family: "Lucida Console", Monaco, monospace;
font-weight: bold;
font-stretch: condensed;
font-style: oblique;
font-variant: normal;
font-size: xx-large; }

h2 { font-family: "Comic Sans MS", cursive;
font-weight: normal;
font-stretch: normal;
font-style: italic;
font-variant: small-caps;
font-size: 20px; }

h3 { font-family: "Palatino Linotype", "Book Antiqua",
                  Palatino, serif;
font-weight: 200;
font-stretch: expanded;
font-style: normal;
font-variant: normal;
font-size: medium; }

h4 { font-family: Arial, Helvetica, sans-serif;
font-weight: 100;
font-stretch: expanded;
font-style: normal;
font-variant: small-caps;
font-size: small; }
```

A fenti CSS kódolás vagy a HTML oldal fej részébe a *<style>….</style>* címkék közé, vagy csatolt CSS fájl esetén a *display.css* (ha ezt a fájlnevet írtuk elő a HTML oldal fej részének *<link……..>* sorában) szövegfájlba írható be.

A böngészőkben a megjelenítés (a vízszintes vonaltól felfelé van a formázott tartalom, alatta pedig az eredeti, formázatlan, a böngészők alapértelmezett értékei szerinti tartalom - ha a CSS csatolt fájl nem érhető el, vagy a stílust megjegyzésbe rakjuk:

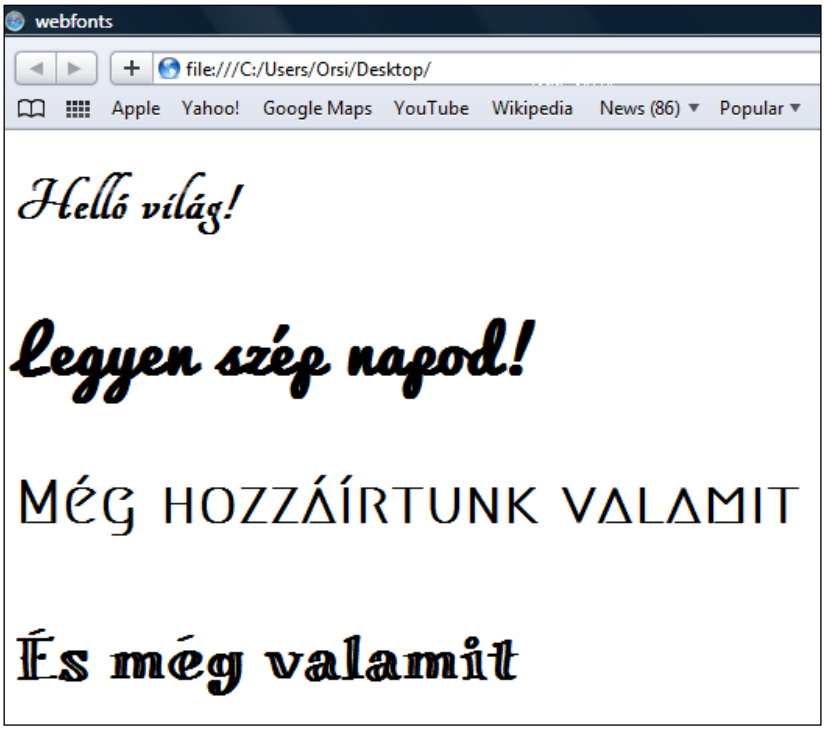


A web-biztos betűtípusok viszonylag csekély száma ösztönzi a webről letölthető *font*-ok használatának terjedését.

a) Először egy olyan, minden böngészővel működő, szabad hozzáférésű megoldást mutatunk be, mely új CSS kódolási ismeretet sem igényel. A letöltendő betűtípusokat a Google *http://google.com/webfonts* oldaláról választjuk ki, azaz ezt használjuk letöltendő stíluslapként, és ezekkel a szokásos négy címünket formázzuk meg.

A fej részben helyezzük el a betűtípusokra mutató *link*-eket, ugyanúgy, mint amikor külső stíluslapokat kapcsolunk a HTML-oldalhoz. A *<style>……</style>* elemben pedig az egyes *h* címsorokhoz rendeljük a kiválasztott betűtípust, tartalékként megadva egy gyűjtő családnevet (ha nem töltődne le az első helyen megadott betűtípus):

```
<meta charset="utf-8">
<title>webfonts</title>

<link rel="stylesheet"
      href="http://fonts.googleapis.com/css?family=Tangerine">
<link rel="stylesheet"
      href="http://fonts.googleapis.com/css?family=Pacifico">
<link rel="stylesheet"
      href="http://fonts.googleapis.com/css?family=Megrim">
<link rel="stylesheet"
      href="http://fonts.googleapis.com/css?family=Miltonian">

<style>
      h1 { font-family:Tangerine, serif; font-size:40px; }
      h2 { font-family:Pacifico, serif; font-size:40px; }
      h3 { font-family:Megrim, serif; font-size:40px; }
      h4 { font-family:Miltonian, serif; font-size:40px; }
</style>
```



Az egyes címsorokat egyforma magasságúra formáztuk, hogy jól látszódjanak a különbségek.

b) A letölthető betű-készletekkel szemben támasztott két fontos követelmény a kis fájlméret és a másolhatatlanság (más dokumentumban vagy alkalmazásban ne legyenek használhatók). A szerzői jogok miatt csak nagyon körültekintően és korlátozottan lehet használni a web-ről letölthető *font*-okat, és a felhasználás szabványosításán is még dolgozik a W3C. A jelenlegi helyzet a következő:

- Apple és Microsoft kidolgozták a True Type formátumot, kiterjesztése *.ttf*
- Microsoft Adobe-val ezt továbbfejlesztette, ez lett az Open Type, kiterjesztése *.otf*
- Microsoft egyedül még tovább fejlesztette, ez lett a EOT (Embedded Open Type), kiterjesztése *.eot*
- Apple is továbbfejlesztette egyedül a True Type-ot, ez lett az ATT (Apple Advanced Technology)

Az EOT-t csak az Internet Explorer-ek, az ATT-t csak a Safari-k értelmezik. A True Type és Open Type viszont gyakorlatilag csereszabatos, és ezekhez szabványosítják az Opera, Microsoft és Mozilla javaslatára a WOFF (Web Open Font Format)-ot, mely egy csökkentett méretű, TT-t és OT-t becsomagoló fájl-formátum. A WOFF-ot már értelmezik az új böngészők: az Internet Explorer 9, az Opera 11.1-től, a Firefox-ok, a Chrome-ok, és a Safari az 5.1 verziótól.

A CSS kódban a *@font-face* szabállyal definiálható a letöltendő betű-készlet, melynek tulajdonságaként a betű-család és annak forrása (a teljes URL) adandó meg:

```
@font-face {
 font-family:…….;
 src: url(http://.....com/fonts/.....fájlkiterjesztés); }
```

A kiterjesztés egyre inkább a *.woff* lesz, mely tehát nem új betűkészlet-formátum, hanem egy becsomagoló fájl-formátum.

Ezek után a szokásos módon a kijelölt elem(ek) tulajdonságaként kezelendő a betűtípus, melynek értékét a fentiekben már megadtuk. Valamennyi bekezdésre vonatkoztatva pl.:

```
p { font-family:........, serif; }
```

ahol a *serif* gyűjtő családnév általános tartalék, ha a megadott *web-font* éppen nem érhető el.

Figyelem! A letölthető *font*-ok csak on-line működnek!

A betű-típusok meghatározására felhasznált CSS tulajdonságok:

font-style betű-stílus
font-variant betű-változat
font-weight betű-vastagság
font-size betű-méret
font-family betű-család
font-stretch betű-kiterjedés
font összevont forma a betű-típus tulajdonságok (*font-stretch* és *font-variant* kivételével történő) megadására

3.9. Színek definiálása

A **color (** szín) tulajdonság és értékének kódolása a *Színek* fejezetben leírtak szerint többféleképpen történhet:

a) RGB színmódban a normál vörös:

color: *#ff0000**;*** (hexa)
color: *#f00**;*** (rövid hexa)
color: *rgb (100%, 0%, 0% **);*** (százalékos)
color: *rgb* (255, *0, 0 **);*** (decimális)
color: *red**;*** (névvel)

A színek értékében szereplő betűk kisbetű/nagybetű érzéketlenek, de célszerű maradni a kisbetűknél.

b) RGBA színmódban a normál vörös:

color: rgba (*100%, 0%, 0%, 1* ***);*** (százalékos)
color: rgba (*255, 0, 0, 1* ***);*** (decimális)

c) RGBA színmódban a vörös, ha nem teljesen fedi az alatta lévő színt:
color: rgba (*100%, 0%, 0%, 0.8* ***);*** (százalékos)
color: rgba (*255, 0, 0, 0.8* ***);*** (decimális)

d) HSL színmódban a normál vörös:
color: hsl (*0, 100%, 50%* ***)*** (csak egy fajta lehetőség van)

e) HSLA színmódban a normál vörös:
color: hsla (*0, 100%, 50%, 1* ***)*** (csak egy fajta lehetőség van)

f) HSLA színmódban a vörös, ha nem teljesen fedi az alatta lévő színt:
color: hsla (*0, 100%, 50%, 0.8* ***)*** (csak egy fajta lehetőség van)

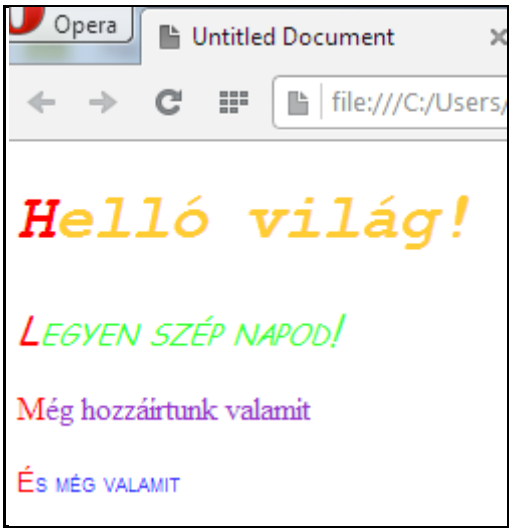
Figyelem! Az Internet Explorer 8 nem értelmezi a HSL/ HSLA színmódot, ezért használata esetén egy hasonló RGB színt is célszerű megadni. Az RGB szín legyen a kódban legelől, így azok a böngészők, melyek értelmezik a HSL-t, a hierarchia miatt azt veszik figyelembe, az IE 8 pedig az RGB-t. (A többi böngészővel már régóta használható a HSL/HSLA színmódban történő színmegadás, és az IE9 is támogatja.)

A *color* (szín) tulajdonság mindig az előtérszínt jelenti (szöveg, keret, stb.). A háttérszínt a *background-color* (háttérszín) tulajdonság határozza meg (lásd a következő fejezetben).

Az előző négy címsoros - immár különböző betű-tulajdonság/értékekkel formázott - HTML oldalunkat színnel is ellátva a korábbi CSS kódoláshoz hozzáírjuk:

```
h1 { color: gold; }
h2 { color: greenyellow; }
h3 { color: dodgerblue; }
h4 { color: palevioletred; }
h1::first-letter, h2::first-letter, h3::first-letter,
h4::first-letter { color: red; }
```

Az így formázott megjelenítés:



Az első betűkre vonatkozó formázási kijelölés szűkebb értelmezésű mint a címsorokra vonatkozó kijelölés, tehát a CSS hierarchiája szerint a szűkebb felülírja az általánosabbat, azaz az első betűk színére adott meghatározás felülírja az adott címsorok színének meghatározását.

3.10. Háttérszín definiálása

A weblap valamennyi elemének (karakter, szó, címsor, sor, bekezdés, lista, táblázat, kép, stb.) van háttérrétege, mely alapértelmezetten teljesen átlátszó. Az ettől eltérő értéket vagy a háttérszín (*background-color*), vagy/és a háttérkép(ek) (*background-image*) tulajdonsággal lehet definiálni (a háttérképekkel külön fejezet foglalkozik).

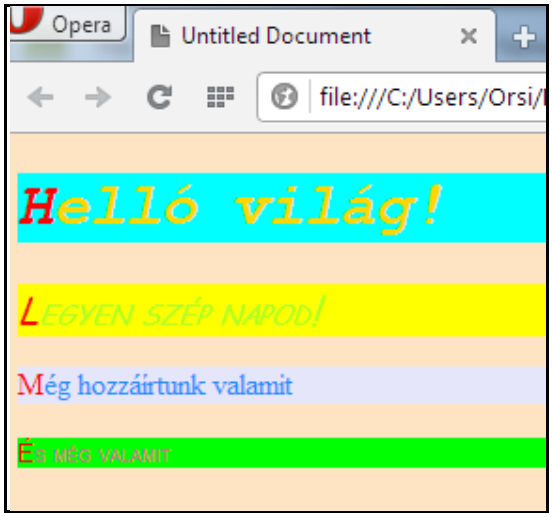
A **background-color** (háttérszín) tulajdonság lehetséges értékei és megadási módjuk megegyezik a *Színek definiálása* fejezetben az előtérszínnél (*color*) ismertetettel, tehát pl. a piros háttérszín RGB-ben az alábbi módokon kódolható:

background-color:*#ff0000;*	(hexa)
background-color:*#f00;*	(rövid hexa)
background-color:*rgb(100%, 0%, 0%);*	(százalékos)
background-color:*rgb (255, 0, 0);*	(decimális)
background-color:*red;*	(névvel)

A négy címsoros, formázott és színezett HTML oldalunkat soronként háttérszínnel is ellátva:

```
body { background-color:bisque; }
   h1 { background-color:aqua; }
   h2 { background-color:yellow; }
   h3 { background-color:lavender; }
   h4 { background-color:lime; }
```

A megjelenítés:



A blokkszintű elemek – jelen esetben az egyes címsorok – dobozát (egyéb beállítás hiányában) a böngésző maga jelöli ki. A doboz függőleges méretét a tartalom magasságához illeszkedve határozza meg, a szélességgel pedig a rendelkezésre álló vízszintes helyet tölti ki. A háttérszínek ennek megfelelően a példánkban a *body* teljes szélességét kitöltik, a függőleges kiterjedésük pedig a címsorok magasságának függvényében változik.

A weboldal törzsének háttérszínét a CSS rangsor alapján takarják a címsorok háttérszínei, azokat pedig a címsorok betűszínei (az első betűkre definiált színek pedig mindegyikkel szemben elsőbbséget élveznek).

<u>A színek és háttérszínek meghatározására felhasznált CSS tulajdonságok:</u>

color	(előtér)szín
background-color	háttérszín

3.11. Blokkszintű elemek elhelyezése I.

A blokkszintű elemek elhelyezése a doboz-modell alapján történik.

3.11.1. Méretek

A **width** (szélesség) tulajdonság lehetséges értékei:

- auto : ez az alapértelmezett érték - a böngésző maga határozza meg egyéb tulajdonságok értékei alapján a szélességet
- abszolút értékben megadás : a tartalom-doboz szélességét definiálja
- relatív értékben (a befoglaló blokk szélességének a százalékában) megadás

Egy bekezdés tartalmának szélességét abszolút értékben (100px) az alábbiak szerint lehet pl. definiálni:

```
p { width: 100px; }
```

Figyelem! Táblázat sorokra és táblázat sor-csoportokra nem használható a *width* tulajdonság.

A **height** (magasság) tulajdonság a *width*-el analóg módon működik, értékei:

- auto : ez az alapértelmezett érték – a böngésző maga határozza meg egyéb tulajdonságok értékei (pl. adott *width*-nél a belső arányok - *intrinsic ratio*) alapján
- abszolút értékben megadás : a tartalom-doboz magasságát definiálja
- relatív értékben (a befoglaló blokk magassága százalékában) megadás

Figyelem! Táblázat oszlopokra és táblázat oszlop-csoportokra nem használható a *height* tulajdonság.

Megjegyzés: Sem a *width*-nek, sem a *height*-nek nem lehet negatív az értéke.

min-width (legkisebb szélesség)**:** az elem legkisebb mérete, nem lehet keskenyebbre venni ennél az értéknél, alapértelmezetten 0 (mind abszolút, mind relatív értékben)

max-width (legnagyobb szélesség): az elem legnagyobb mérete, nem lehet szélesebbre venni ennél az értéknél, alapértelmezetten *none* – azaz ha nem adjuk meg, az egyéb tulajdonságoktól függ a szélesség (mind abszolút, mind relatív értékben)

min-height (legkisebb magasság): az elem legkisebb mérete, nem lehet alacsonyabbra venni ennél az értéknél, alapértelmezetten 0 (mind abszolút, mind relatív értékben)

max-height (legnagyobb magasság): az elem legnagyobb mérete, nem lehet magasabbra venni ennél az értéknél, alapértelmezetten *none* – azaz ha nem adjuk meg, az egyéb tulajdonságoktól függ a magasság (mind abszolút, mind relatív értékben).

Figyelem! Táblázat-sorokra, táblázat sor-csoportokra, táblázat-oszlopokra és táblázat oszlop-csoportokra nem érvényesek a *min*- és *max*- tulajdonságok.

Megjegyzés: Sem a *min*- sem a *max*- tulajdonságoknak nem lehet negatív az értéke.

3.11.2. Elhelyezés

3.11.2.1. A belső margó (*padding* - lásd az elemi doboz-modell fejezetet) megadásának módja a *padding-top*, (felső belső margó), *padding-right* (jobboldali belső margó), *padding-bottom* (alsó belső margó), *padding-left* (baloldali belső margó) tulajdonságokkal történik. Alapértelmezett értékük 0 (mind abszolút, mind relatív értékben) és egyiknek sem lehet negatív az értéke.

A fenti értékek összevontan is megadhatók a *padding* tulajdonsággal, a felsorolt értékek kötelező sorrendje az óramutató járásával megegyezik és felülről indul. Ha a bal értéket kihagyjuk, az megegyezik a jobbal, ha az alsó értéket hagyjuk ki, az megegyezik a felsővel. Ha mindegyik érték megegyezik, egyetlen rövid alakú *padding* tulajdonság használható.

Tehát pl. ekvivalens megadás:

```
h1 { padding: 0.5em; }

h1 { padding-top: 0.5em;
     padding-right: 0.5em;
     padding-bottom: 0.5em;
     padding-left: 0.5em; }
```

A belső margó megadható abszolút értékben (*px*, *em*) vagy százalékosan. Az alsó és felső belső margó százalékos értéke is a szélességhez viszonyított, nem a magassághoz!

3.11.2.2. A külső margó (*margin* - lásd az elemi doboz-modell fejezetet) tulajdonsággal a befoglaló blokk-elemnek (az ő befoglaló blokkjához képesti) helyzete adható meg. A *margin-top*, *margin-right*, *margin-bottom*, *margin-left*, ill. összevont *margin* alakjuk a belső margónál (*padding*) leírtakkal analóg módon működik.

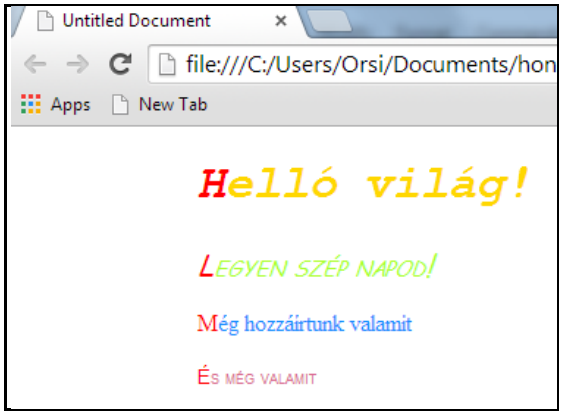
Alapértelmezett értékük szintén 0 (mind abszolút, mind relatív értékben). Lényeges különbség a belső margóhoz képest, hogy a külső margónak negatív értéke is lehet - ekkor a befoglaló blokkból kilóg az elem, és akár egy másik blokkal átfedésbe is kerülhet.

A weblap törzsét pl. középre igazítva, mindkét oldalán 10% - 10%-os margót hagyva:

```
body { margin-right:10%; margin-left:10%; }        vagy
body { width:80%; margin-right:auto; margin-left:auto; }
```

Az egymást követő blokkszintű elemek – jelen esetben a címsorok – alapvetően egymás alatt helyezkednek el, és bár helyzetük a fejezetben ismertetett tulajdonságokkal változtatható, sorrendiségükből kiemelni csak a *Blokkszintű elemek elhelyezése II.* fejezetben tárgyalt technikákkal lehet.

A címsorok megjelenítése (a weblap természetesen jobbra még folytatódik):



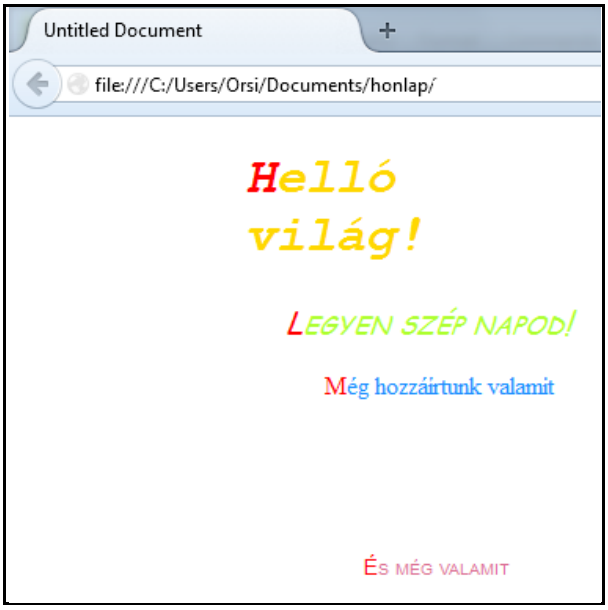
A négy címsor dobozát *25px*-el mindig jobbra tolva, a *h1* címsorra max. *50px* szélességet, a *h3*-ra *100px* magasságot definiálva a kódolás:

```
h1 { max-width:115px; margin-left:25px; }
h2 { margin-left:50px; }
h3 { height:100px; margin-left:75px; }
h4 { margin-left:100px; }
```

A *25px*-enkénti jobbra tolás mellett látható, hogy:

- a *h1* tartalom doboz szélességének korlátozásával az adott betű-jellemzőkkel a szöveg egy sorban nem fér el a tartalom-dobozban, ezért két sorba bontva jeleníti meg a böngésző
- a *h3* tartalom doboz magasságát megnövelve a *h4* tartalom doboza a megemelt magasságnak megfelelően lejjebb került

A megjelenítés:



<u>A blokkszintű elemek méretezésére és elhelyezésére felhasznált CSS tulajdonságok:</u>

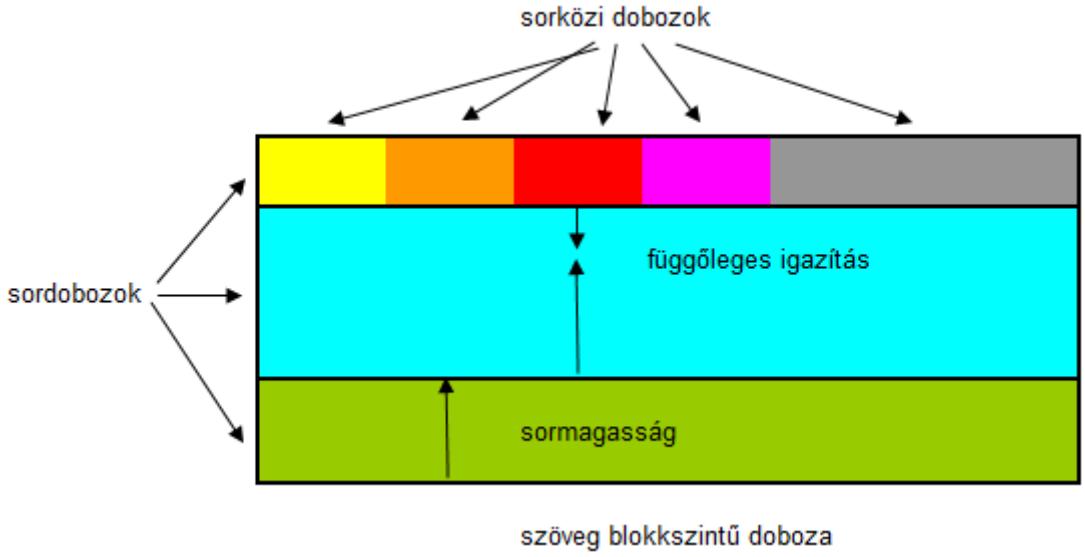
min-width, max-width, width	(minimális, maximális) szélesség
min-height, max-height, height	(minimális, maximális) magasság
padding, padding-top, -right, -bottom, -left	(felső, jobb, alsó, bal) belső margó
margin, margin-top, -right, -bottom, -left	(felső, jobb, alsó, bal) külső margó

3.12. Szöveg formázása

Ez a fejezet a vízszintes sorokból álló, egyhasábos szövegek formázásának technikáját mutatja be. (A vízszintes sorokból álló, többhasábos elrendezésű szöveg kialakítása külön fejezetben kerül tárgyalásra.)

A szövegre vonatkozó tulajdonságokkal a pozícionálás, a karakterek, a betűköz, a szóköz, a szótörés, az elválasztás és az árnyékolt szöveg megjelenítése definiálható.

A pozícionáláshoz az elemi doboz-modellnél már ismertetett sor-doboz és sorközi doboz fogalmak adnak támpontot:



3.12.1. Szöveg pozícionálása és díszítése

A **text-indent** (szöveg behúzása) tulajdonság egy szövegblokk első sorának behúzását definiálja. A doboz-modell alapján a szöveg blokkszintű doboza első sor-dobozában az első sorközi doboz helyzetét határozza meg. A sor-doboz bal szélétől számítandó a behúzás, amit a böngészők üres hellyel töltenek ki. Értéke megadható fix számmal vagy a befoglaló doboz szélességéhez viszonyítva százalékosan.

Példa egy bekezdés első sorának *3em* behúzására:

```
p { text-indent:3em; }
```

A **text-align** (szöveg igazítása) tulajdonság egy szövegblokk adott sor-dobozában lévő sorközi dobozok helyzetét adja meg, lehetséges értékei: *left* (bal), *right* (jobb), *center* (közép), *justify* (sorkizárt, jusztírozott).

A left, right, center értékek esetében a tulajdonság azt specifikálja, hogy az egyes sor-dobozokban a sorközi dobozok hogyan helyezkednek el a sor-doboz bal és jobb széléhez képest. (A sorkizárt esetben viszont nemcsak a sorközi dobozok helyzetét, de még magukat a sorközi dobozokat is módosítja a formázás.)

Példa egy sorkizártan formázott szövegblokkra:

```
div { text-align:justify; }
```

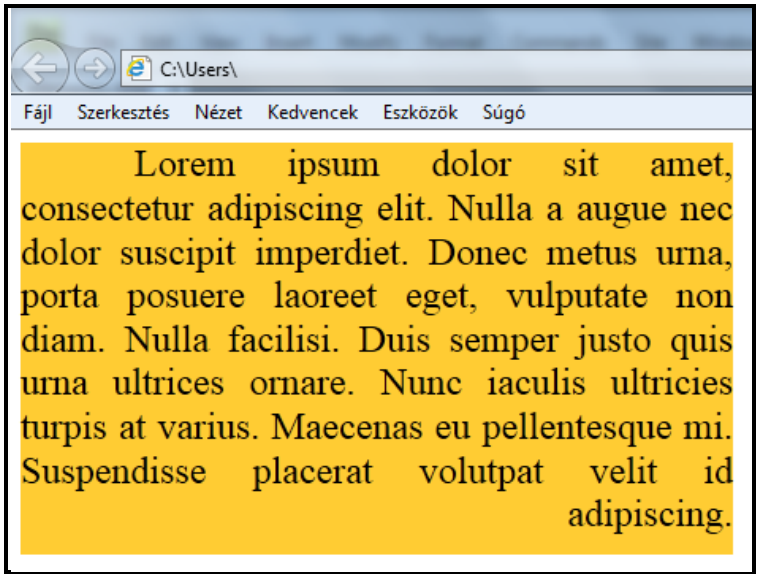
Megjegyzés: Amíg a böngészők nem értelmezik általánosan a magyar helyesírás szerinti automatikus szóelválasztást, addig a sorkizárt elrendezés – különösen keskeny sorok esetén - nagyon egyenetlen szövegoszlopot eredményezhet.

A **text-align-last** (szöveg utolsó sorának igazítása) tulajdonság egy sorkizárt (*text-align: justify* tulajdonság/értékű) szövegblokk utolsó, nem teljes sorának igazítását határozza meg. Lehetséges értékei a *text-align* értékeihez hasonlóan *left, right* és *center*.

Példa egy sorkizárt szövegblokk utolsó sorának jobbra igazítására:

```
div { text-align:justify; text-align-last:right; }
```

Egy *3em* szövegbehúzású, sorkizárt szövegblokk megjelenítése, ha az utolsó (nem teljes) sor jobbra igazított:

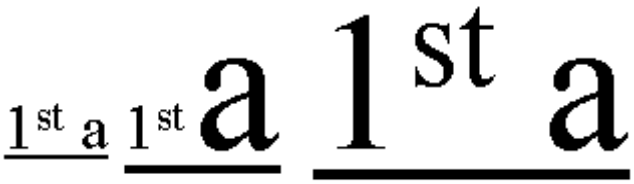


Megjegyzés: A *text-alig-last* tulajdonságot csak az Internet Explorer-ek és (*-moz-* előtaggal) a Firefox értelmezi, a többi böngésző nem veszi figyelembe a kódolást és mindig balra igazít.

A **text-decoration** (szöveg díszítése) tulajdonság vonalakkal díszíti a szöveget. A lehetséges értékei:

none	nincs semmilyen szövegdekoráció – ez az alapértelmezett
underline	a szöveg minden sora alatt van vonal
overline	a szöveg minden sora felett van vonal
line-through	a szöveg minden sora a közepén vonallal áthúzott

A szövegdíszítés az aláhúzást, áthúzást, felülhúzást csak szövegre alkalmazza, beleértve a betű- és szóközöket is – a szegélyt, belső és külső margót átugorják. A vonalak pozícióját és vastagságát a böngészők határozzák meg (pl. a betűméret, betűtípus, stb. alapján), ezek kódolással nem állíthatók. A vonalak színe a szöveg fő színével egyezik meg:



Megjegyzés: Korábban a szöveg villogása (*blink* - látható és nem látható állapot közötti váltakozása) is szerepelt a szabványban, de csak a Firefox és az Opera jelenítették meg régebbi verzióikban. Frusztráló hatása miatt nem népszerű tulajdonság (az új böngészők közül már egyik sem hajtja végre, a Chrome, Internet Explorer és Safari-ba szándékosan soha nem is építették be ezt a funkciót), aki mégis alkalmazni kívánja, az *Animáció* fejezetben talál rá megoldást.

A **text-transform** (kisbetű/nagybetű átalakítás) tulajdonság lehetséges értékei:

- *none* (alapértelmezett): nincsen nagybetű hatás
- *capitalize*: minden szó első karakterét nagybetűre változtatja, a többi karaktert nem érinti
- *uppercase*: minden szó minden karakterét nagybetűsre változtatja
- *lowercase*: minden szó minden karakterét kisbetűsre változtatja

Például egy címsort teljesen nagybetűssé alakítva:

```
h1 { text-transform:uppercase; }
```

A **letter-spacing** (betűköz) tulajdonság a szöveg karakterei közötti térközt adja meg. Lehetséges értékei:

- *normal* (alapértelmezett): az adott betűtípushoz tartozó normál betű-köz – ez az érték lehetővé teszi a böngészők számára a karakterek közti térköz változtatását sorkizárt elrendezés kialakításakor
- *length* (távolság értéke): az alapértelmezett betűközhöz képesti változást, tehát nem a betűköz abszolút értékét adja meg. Mindkét – tehát pozitív és negatív – irányban is értelmezhető. Rögzített értékről lévén szó, a böngészők nem tudják változtatni (növelni vagy csökkenteni) a betűközt a sorkizárt elrendezés érdekében.

Például a betűközt *0.1em* -el növelve:

```
blockquote { letter-spacing:0.1em; }
```

Azért is rögzíthető a távolság, hogy a böngésző ne változtathassa meg a betűközt, pl.:

```
blockquote { letter-spacing:0; }
```

A **word-spacing** (szóköz) tulajdonság a szavak közötti elválasztó karakterközt (szóközt) specifikálja, lehetséges értékei:

- *normal* (alapértelmezett): az adott betűtípushoz tartozó szóköz
- *length* (távolság értéke): az alapértelmezett szóközhöz képesti változást, tehát nem a szóköz abszolút értékét adja meg. Mindkét – tehát pozitív és negatív – irányban is értelmezhető (a sorkizárt elrendezés is befolyásolja).

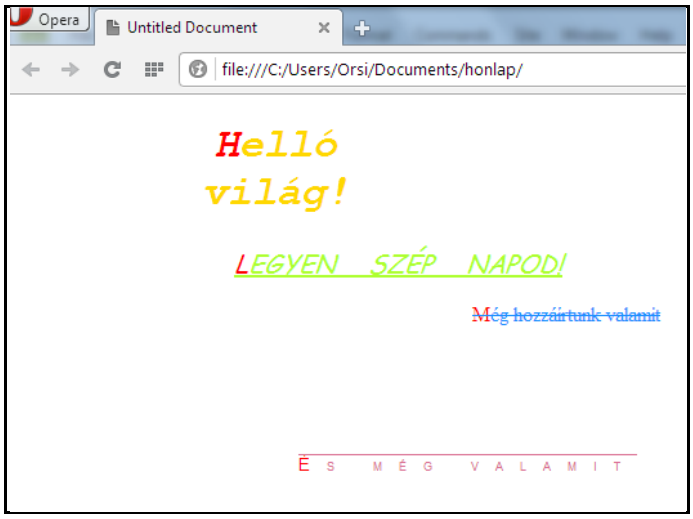
Például a szóközt *1em*-el növelve:

```
h1 { word-spacing:1em; }
```

A fenti tulajdonságok bemutatására folytatjuk a négy címsoros szöveg formázását:

```
h1 { text-decoration:none; text-align:center; }
h2 { text-decoration:underline; text-transform:uppercase;
     word-spacing:1em; }
h3 { text-decoration:line-through; text-indent:10em; }
h4 { text-decoration:overline; letter-spacing:1em; }
```

A megjelenítés:



A megjelenítésből megállapítható, hogy:

- az alá-, fölé-, és áthúzás színe egy szövegdíszítésen belül akkor sem változik, ha a karakterek egy részének változik a színe - a vonalak a szöveg fő színével egyeznek meg
- az alá-, fölé-, és áthúzás a betűközöket és szóközöket is folyamatosan áthidalja

A szövegdíszítési tulajdonságok igazából csak korlátozottan alkalmazhatóak, mert:

a) az aláhúzásról elsősorban a hivatkozásra asszociálnak a felhasználók, ha pedig tartalmi oka van, arra az <*u*> és <*ins*> HTML címkéket célszerű használni

b) ha a szöveg áthúzásának tartalmi oka van, arra szolgálnak az <*s*> és <*del*> HTML címkék (tisztán díszítésnek az áthúzás kevéssé használatos)

c) a felülhúzott szövegdíszítés még ritkábban használható

Azonban a *text-decoration* tulajdonság továbbfejlesztés alatt áll, melynek során egyes paraméterei külön-külön is definiálhatókká válnának:

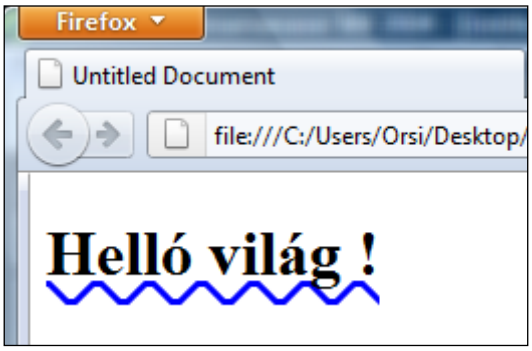
- *text-decoration-line* (szövegdíszítő vonal) tulajdonság (régi) értékei adják meg a vonal helyzetét (*none, underline, overline, line-through*)
- *text-decoration-color* (szövegdíszítés színe) a díszített szöveg színétől független, tetszőleges vonalszín definiálását teszi lehetővé
- *text-decoration-style* (szövegdíszítés stílusa) a vonal jellegét definiálja, lehetséges értékei: *solid* (folytonos), *dotted* (pontozott), *dashed* (szaggatott), *double* (dupla) és *wavy* (hullámos)
- *text-decoration-skip* (szövegdíszítő vonal ugrás/kihagyás), lehetséges értékei: *none* (nincs kihagyás, mindent folytonosan aláhúz), *images* (sorban elhelyezett képek kihagyása/átugrása), *spaces* (elválasztó/üres karakterek kihagyása/átugrása), *ink* (a karakterek rajzolatának kihagyása/át-ugrása).

Az eredeti *text-decoration* összevont tulajdonsággá válna, melynek tagjai (kötött sorrendben) a *text-decoration-line, text-decoration-color, text-decoration-style* és *text-decoration-skip* lennének – ha nincs megadva *text-decoration-color, text-decoration-style* és *text-decoration-skip* érték, a *text-decoration* tulajdonság visszafelé kompatibilis maradna a korábbi szabvánnyal.

Megjegyzés: Az új *text-decoration-*……tulajdonságokkal a szövegdíszítésre vonatkozó korábbi negatív megállapítások részben módosulnak, de egyelőre csak a Firefox (*-moz-* előtaggal és a *skip* kivételével) értelmezi az újdonságokat.

Példa az új szövegdíszítési lehetőségekre:

```
h1 { -moz-text-decoration-line:underline;
     -moz-text-decoration-color:blue;
     -moz-text-decoration-style:wavy;
     -moz-text-decoration-skip:spaces; }
```



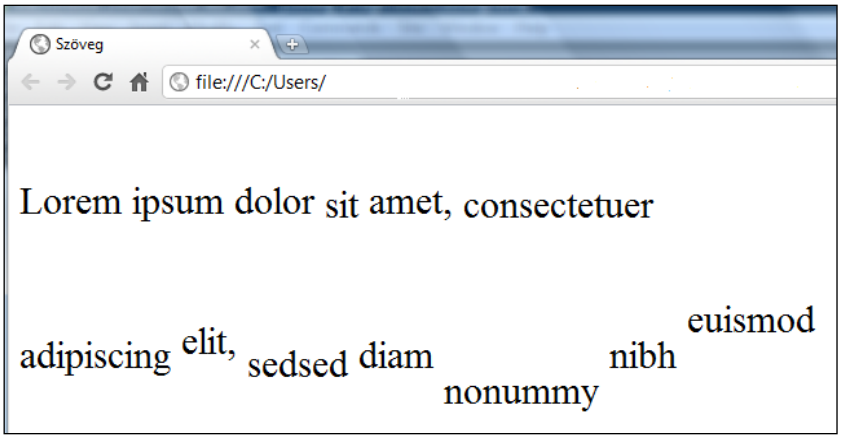
A **line-height** (sormagasság - kitöltésnek is nevezik) a betűmérethez viszonyítva adható meg, lehetséges értékei:

- *normal*: ez az alapértelmezett, a böngésző szabja meg a betűméret alapján (javasolt értéke kb. 1,0 - 1,2)
- százalékosan: pl. *line-height: 120%; font-size: ……;*
- egy (pozitív) szorzószámmal: pl. *line-height: 1.2; font-size: ……;*
- *em*-ben: pl. *line-height: 1.2em; font-size: ……;*

A **vertical-align** (függőleges igazítás) tulajdonság a soron (a sorközi dobozoknak a sordobozon) belüli függőleges irányú elhelyezkedést definiálja, lehetséges értékei: *top* (fent), *middle* (középen), *bottom* (lent), *super* (felső index), *sub* (alsó index), *text-top* (szöveg tetején), *text-bottom* (szöveg alján), *central* (középen), *baseline* (betűvonalhoz).

Példa a sormagasság és a függőleges igazítás tulajdonságok definiálására:

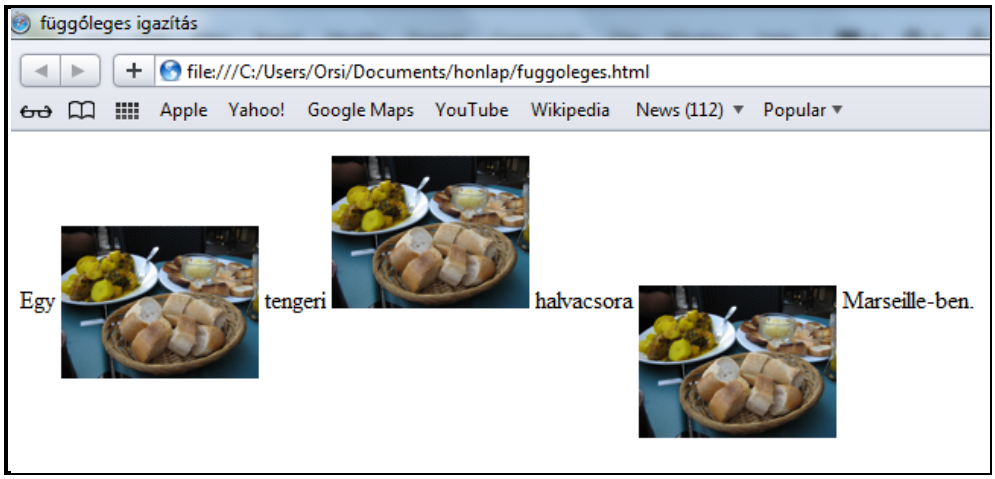
```
<style>
  p #top { vertical-align:top; }
  p #middle{ vertical-align:middle; }
  p #bottom { vertical-align:bottom; }
  p #super { vertical-align:super; }
  p #sub { vertical-align:sub; }
  p #text-top { vertical-align:text-top; }
  p #text-bottom { vertical-align:text-bottom; }
  p { line-height:3em; font-size:30px; }
</style>

<p>Lorem <span id="top">ipsum</span> dolor <span id="middle">
sit</span> amet, <span id="bottom">consectetuer</span><br>
adipiscing <span id="super">elit,</span><span id="sub">
 sedsed</span> diam <span id="text-top">nonummy</span> nibh
<span id="text-bottom">euismod</span>
</p>
```



Gyakoribb a *vertical-align* tulajdonság használata szöveg és kép, beviteli mező, stb. egymáshoz igazítására. Margó nélküli kép betűvonalba illesztésekor az alapértelmezett *baseline*, margóval ellátottnál a *bottom*, középre igazítottnál a *middle* használatos, stb.

Pl. a korábbi halvacsorás kép különböző függőleges irányú elhelyezése a szövegben:

```
<style>
#middle { vertical-align: middle; }
#bottomline { vertical-align: bottomline; }
#text-top { vertical-align: text-top; }
img { width: 130px; height: 100px; }
</style>

<p>Egy <img src="IMG_1937.JPG" id="middle"> tengeri
    <img src="IMG_1937.JPG" id="bottomline"> halvacsora
    <img src="IMG_1937.JPG" id="text-top"> Marseille-ben.
</p>
```



3.12.2. Szöveg tördelése és elválasztása

A **white-space** (elválasztó karakterek kezelése) tulajdonsággal az a HTML-szabály finomhangolható, miszerint a kódolás mindig csak **egy** elválasztó karaktert vesz figyelembe.

Az elválasztó karakterek egy elemen belüli kezelésének a lehetséges értékei:

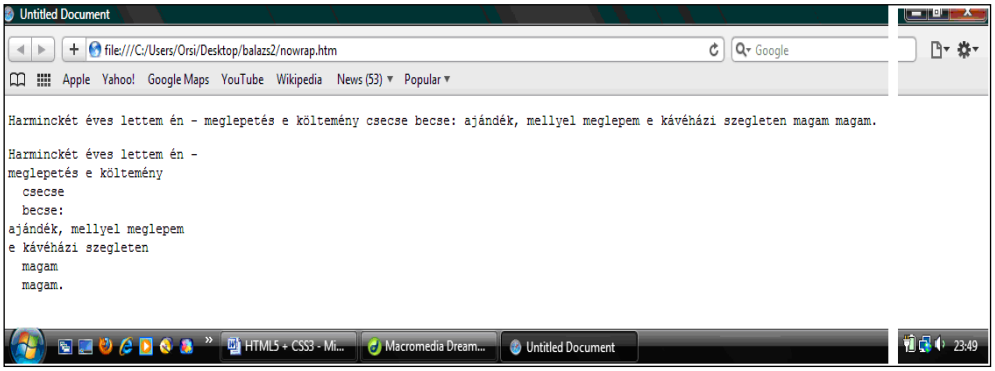
- *normal* (alapértelmezett): a szomszédos elválasztó karaktereket összevonja, a sortöréseket a sor-dobozok kitöltésének megfelelően hajtja végre

- *pre* (előformázott): megakadályozza a szomszédos elválasztó karakterek összevonását, a sortöréseket a kódolásban elhelyezett „új sor”-nál hajtja végre
- *nowrap* (nem csomagol össze): *normal*-ként összevonja a szomszédos elválasztó karaktereket, de elnyomja a szövegben lévő sortöréseket
- *pre-wrap*: megakadályozza a szomszédos elválasztó karakterek összevonását, a sortörést a kódolásban elhelyezett „új sor”-nál, ill. a sor-dobozok kitöltésének megfelelően végzi
- *pre-line*: *normal*-ként összevonja a szomszédos elválasztó karaktereket, a sortörést a kódolásban elhelyezett „új sor”-nál, ill. a sor-dobozok kitöltésének megfelelően végzi

Például egy táblázat celláihoz:

```
td, th { white-space:nowrap; }
```

A *white-space* tulajdonság *nowrap* értékének hatása az előformázásnál már idézett József Attila „Születésnapomra” c. versének első két versszakán megfigyelhető - a CSS-ben a ***pre { white-space:****normal****; }*** megőrzi, a ***pre { white-space:****nowrap****; }*** elnyomja a szövegben lévő sortöréseket:



Az automatikus szóelválasztás tulajdonsága a **hyphens**, lehetséges értékei *manual* (ez az alapértelmezett), *none* és *auto*. Ebben az esetben a HTML dokumentumban feltétlenül definiálni kell a *<html lang=”..”>* főnyelvet, hogy egyértelmű legyen a böngészőnek, melyik nyelv elválasztási szabályait kell alkalmaznia. Kísérleti stádiumban van ez a tulajdonság, jelenleg az Internet Explorer, Firefox és Safari próbálkozik vele saját (*-ms-*, *-moz-*, *-webkit-*) előtaggal, a Chrome és Opera megkezdte a bevezetését, majd felhagyott vele. A magyar nyelvű elválasztást az IE és Safari sem támogatja, tehát (még) nem használható tulajdonság.

3.12.3. Szöveg árnyéka

A **text-shadow** (szöveg árnyéka) tulajdonsággal egy vagy több, különböző színű, nagyságú, irányú és életlenítési/elhalványulási távolságú árnyék rendelhető egy szöveghez.

Az árnyék(ok) az esetleges szövegdíszítést is követi(k), és elfedi(k) a háttérszínt és/vagy háttérképet (ha van ilyen), de nem fedik el a szöveget, ha arra vetülnek. Több árnyék egyidejű alkalmazása esetén az első árnyék van legfelül, és takarja a sorban következő(ke)t.

A *text-shadow* tulajdonsághoz - **egy** árnyék esetén - a szabvány 3 db hosszúság érték és egy szín definiálását teszi lehetővé. A vízszintes és függőleges hosszúság értékeket kötelező megadni, akkor is, ha 0 értékűek, az életlenítési/elhalványulási távolság opcionális – ha nulla, akkor elhagyható a kódolás során.

Egy árnyék érték-komponensei tehát az alábbiak:

- az első hosszúság érték az árnyék vízszintes kiterjedése: pozitív értéknél a szövegtől jobbra, negatív értéknél a szövegtől balra rajzolódik ki az árnyék
- a második hosszúság érték az árnyék függőleges kiterjedése: pozitív értéknél lefelé, negatív értéknél felfelé rajzolódik ki az árnyék
- a harmadik hosszúság érték az életlenítési (*blur*) távolság: negatív érték nem megengedett, ha 0 az értéke, akkor az árnyék széle éles, egyébként pedig minél nagyobb pozitív érték, annál életlenebb az árnyék széle
- az árnyék színe az értékek sorában az utolsó, a színeknél ismertetett módokon definiálható.

Több árnyék esetén az egyes árnyékok értékeinek vesszővel elválasztott listájából áll a többes árnyék definiálása.

Megjegyzés: A *text-shadow* tulajdonság a *::first-letter* és *::first-line* pszeudo-elemekkel is használható.

Valamennyi szöveg-árnyék variációt be tudjuk a sokat használt négy címsoron mutatni az alábbi CSS kódolással (a formázatlan HTML oldal a kiindulópont):

```
h1 { text-shadow: 20px 20px 5px red; }
```

egy árnyék van ferdén lefelé, a színe RGB-ben, színnévvel van megadva

```
h2 { text-shadow: 0px 20px 5px red, 10px 10px 5px hsl(120,
                  100%, 50%); }
```

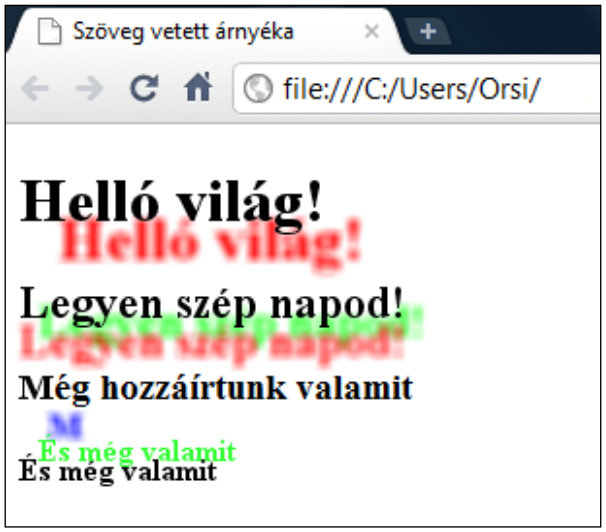
két különböző színű árnyék van lefelé, az egyik függőlegesen, a másik ferdén - részben átfedik egymást, értékeiket vesszővel elválasztva egy tulajdonságon belül soroltuk fel, az egyik szín RGB-ben színnévvel, a másik HSL-ben van megadva

```
h3::first-letter { text-shadow:15px 20px 5px rgb(0, 0, 255); }
```

az árnyék ferdén lefelé vetődik, az *első betű* pszeudo-elemhez tartozik, színe decimális RGB-ben van megadva

```
h4 { text-shadow: 10px -10px hsla(120, 100%, 50%, 0.8); }
```

az árnyék felfelé vetődik, nincsen életlenítése, részben áttetsző színe HSLA-ban van megadva



Figyelem! Az Internet Explorer 8 és 9 nem értelmezik a *text-shadow* tulajdonságot, árnyék nélkül jelenítik meg a szöveget.

Példák a szöveg-árnyék néhány gyakorlati alkalmazására (a főcímünkön bemutatva):

a) Süllyesztett szegélyű (*inset*, *letterpress*) szöveg:

Függőleges irányban egy kis árnyék finom kiemelés benyomását kelti. Az árnyék színének világosság szempontjából a háttér és a szöveg világossága közé kell esnie.

```
body { background-color:#eee; }
h1 { font-size:160px; color:#504f4f; text-shadow:0px 5px 5px
     #bbbaba; }
```

b) Retro:

Kétféle, azonos dőlési szögű árnyék, melyek közül az első színe megegyezik a háttér színével, a második árnyék színe pedig a szövegével.

```
body { background-color:#eee; }
h1 { font-size:160px; color:#504f4f; text-shadow: 5px 5px 0px
     #eee, 7px 7px 0px #707070; }
```

c) Társasjáték:

A szövegtől egyenletesen növekvő távolságra két szín váltakozásával létrehozott, életlenítés nélküli négy árnyék.

```
body { background-color:#eee; }
h1 { font-size:160px; color:#fff; text-shadow: 6px 6px 0px
     #ffd217, 12px 12px 0px #5ac7ff,18px 18px 0px #ffd217,
     24px 24px 0px #5ac7ff; }
```

d) Lángoló szöveg:

Hét árnyék különböző dőlési szöggel és életlenítéssel rendelkezik, színskálájuk pedig a vakító fehértől a sárga, narancs és sötétebb borostyán árnyalatok felé megy át lángok benyomását keltve.

```
body { background-color:#000; }
h1 { font-size:160px; color:#fff; text-shadow:0px 0px 20px
     #fefcc9, 10px -10px 30px #feec85, -20px -20px 40px
     #ffae34, 20px -40px 50px #ec760c, -20px -60px 60px
     #cd4606, 0 -80px 70px #973716, 10px -90px 80px #451b0e; }
```

e) Neon:

A szövegtől távolodva egyre nagyobb életlenítésű és a fehérből lilás színárnyalatba átmenő nyolc árnyék.

```
body { background-color:#000; }
h1 { font-size:160px; color:#fff; text-shadow: 0 0 10px #fff,
     0 0 20px #fff, 0 0 30px #fff, 0 0 40px #ff00de, 0 0 70px
     #ff00de, 0 0 80px #ff00de, 0 0 100px #ff00de, 0 0 150px
     #ff00de; }
```

A szöveg formázására felhasznált CSS tulajdonságok:

text-indent	szöveg behúzásas
text-align	szöveg igazítása
text-align-last	sorkizárt szöveg utolsó sorának igazítása
text-decoration	szöveg díszítése
text-decoration-line	szövegdíszítő vonal
text-decoration-color	szövegdíszítő vonal színe
text-decoration-style	szövegdíszítő vonal stílusa
text-decoration-skip	szövegdíszítő vonal folytonossága
text-transform	kisbetű/nagybetű átalakítás
letter-spacing	betűköz
word-spacing	szóköz
white-space	elválasztó karakterek kezelése
text-shadow	szöveg vetett árnyéka
vertical-align	függőleges igazítás
line-height	sormagasság
hyphens	szóelválasztás

3.13. Listák formázása

A listák formázására a CSS által generált doboz egy fő/tartalmi dobozból (*principal box*) és egy felsorolásjel-dobozból - más fordításban jelölő-dobozból (*marker box*) - áll.

A lista tulajdonságok a felsorolásjel (*marker*) típusát (kép, szám, karakterjel), és a felsorolásjelnek a tartalmi dobozhoz viszonyított helyzetét (kívül vagy belül helyezkedik el) definiálják.

A **list-style-type** (lista stílusának típusa) a felsorolásjelek típusát adja meg, melyek lehetnek számok, betűk, karakterek, ill. *none* (nincsen semmilyen felsorolásjel).

A számozás lehetséges formái (a *list-style-type* értékei):

decimal	tízes számrendszerű 1-től kezdődően
decimal-leading-zero	10 alatt o-val kezdődő tízes számrendszerű (01, 02, 03, ..., 98, 99)
lower-roman	kisbetűs római számok (i, ii, iii, iv, v, stb.)
upper-roman	nagybetűs római számok (I, II, III, IV, V, stb.)
georgian	hagyományos grúz számozás (ez leginkább díszítőelemnek jó)
armenian	hagyományos örmény számozás („)

Nem ismerik fel a böngészők a *circled-decimal* (tízes számrendszerű számok körbe foglalva), *parenthesised-decimal* (tízes számrendszerű számok zárójelbe foglalva) és még további, a szabványban egyébként szereplő számozási lehetőségeket.

A betűk fajtái (a *list-style-type* értékei):

lower-latin vagy **lower-alpha**	latin kisbetűk	(a, b, c, ... z)
upper-latin vagy **upper-alpha**	latin nagybetűk	(A, B, C, ... Z)
lower-greek	görög kisbetűk	(α, β, γ, ...)

A Firefox és Chrome értelmezi a japán szótagjeleket (*kana*-kat) és a héber karaktereket is (*hiragana, katakana* és *hebrew* értékekkel).

Nem ismerik fel a böngészők a *circled-lower-latin, circled-upper-latin* (latin kis- és nagybetűk körbe foglalva), *parenthesised-lower-latin* (latin kisbetűk zárójelbe foglalva), *upper-greek* (görög nagybetűk) és még további, a szabványban egyébként szereplő betű-felsorolásjeleket.

Megjegyzés: A specifikáció nem ad útmutatást arra, hogy mi történjen az abc végén. Tehát 26-nál hosszabb felsorolás esetén pl. a *lower-latin* definiálatlan – ilyen esetben inkább a számozás használata javasolt.

A megadható karakterjelek (a *list-style-type értékei*):

disc (kitöltött kör) - ezt régebben *k*-val írták, mindkét verziót felismerik a böngészők
circle (üres kör)
square (kitöltött négyzet)

Nem ismerik fel a böngészők a *hyphen* (gondolat/bekezdésjel, „bajusz"), a *diamond* (gyémánt), a *check* („pipa" jel) és a *box* (üres négyzet) szabványos karakterjeleket felsorolásjelként.

Megjegyzés: Értelemszerűen a számok és betűk a számozott listáknál, a karakterjelek a számozatlanoknál használatosak, de a számozott listához is megadhatók karakterjelek (ekkor úgy néz ki, mint egy felsorolási lista), és a felsorolási listához is megadható számozás (ekkor úgy néz ki, mintha számozott lenne).

Példa egy görög kisbetűkkel jelölt számozott listára:

```
ol { list-style-type:lower-greek; }
```

A **list-style-image** (felsorolásjel képe) tulajdonsággal egy általunk megadott kép alkalmazható felsorolásjelként. Értelemszerűen elsősorban számozatlan listáknál használatos.

Ha a felsorolásjelnek választott képnek van saját belső (*intrinsic*) magassága/szélessége, akkor ez marad felsorolásjelként is a mérete. Ha a méret százalékosan van megadva, akkor *1em*-hez viszonyított méretben lesz felsorolásjel.

Ha a képet a weboldal gyűjtőmappájában tároljuk, elegendő a kép fájlnevére és kiterjesztésére hivatkozni. Például egy gyémántot ábrázoló, GIF fájltípusú kép felsorolásjelként való alkalmazása esetén:

```
ul { list-style-image:url(diamond.gif); }
```

Ha a *list-style-image* értéke *none* - vagy a kép nem jeleníthető meg - akkor a *list-style-type* értéke fog látszódni.

A **list-style-position** (felsorolásjel helyzete) tulajdonsággal a felsorolásjel-doboz helyzete adható meg a tartalmi dobozhoz képest. Két lehetséges értéke: *outside* (kilógó) vagy *inside* (kilógás nélküli). Például a kilógás nélküli listaelemek kódolása:

```
ol { list-style-position:inside; }
```

A különböző *list-style-….* tulajdonságok *list-style* -ként összevontan is megadhatók az alábbi kötött sorrendben: *list-style-type, list-style-image, list-style-position*. Az értékek között

van üres betűköz, de nincsen vessző. Tehát a fenti példák összevont kódolása, ha nincsen felsorolásjelként alkalmazott kép:

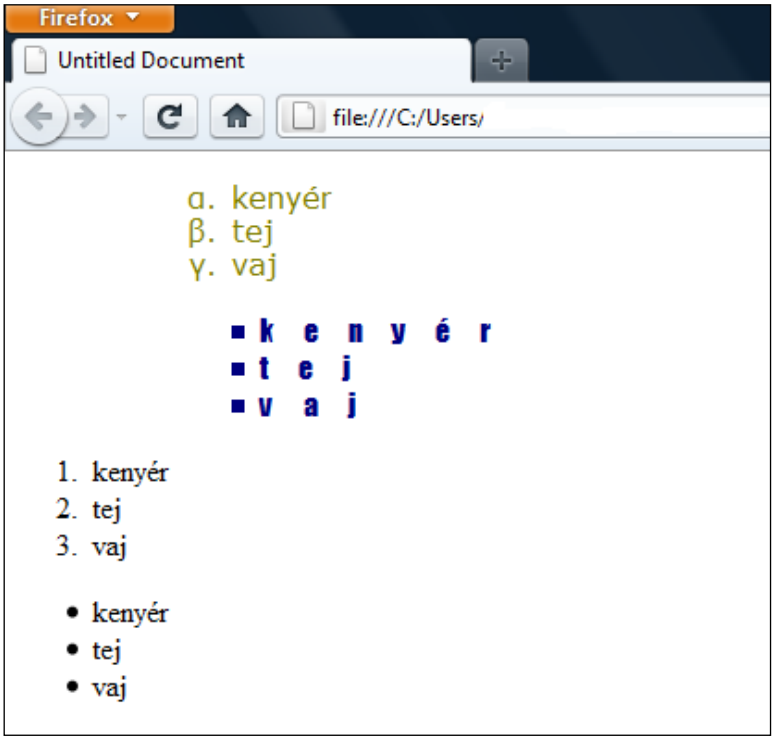
```
ol { list-style:lower-greek inside; }
```

Figyelem! Az *ul { list-style: none }* összevont tulajdonság felülírja az esetleg külön megadott *list-style-image* és *list-style-position* tulajdonságokat, és nem jelenik meg felsorolásjel.

HTML-es élelmiszeres listánkat formázva – pl. új betűtípusokat, betűközöket és színeket alkalmazva, továbbá a listákat az oldal szélétől beljebb tolva – a CSS-kódolás:

```
ol { color:olive; list-style-type:lower-greek; font-family:
     Verdana, Geneva, sans-serif; list-style-position:
     outside; }
ul { color:navy; font-family:Impact, Charcoal, sans-serif;
     letter-spacing:1em; list-style-type:square; list-style-
     position:inside; }
ol, ul { margin-left:75px; }
```

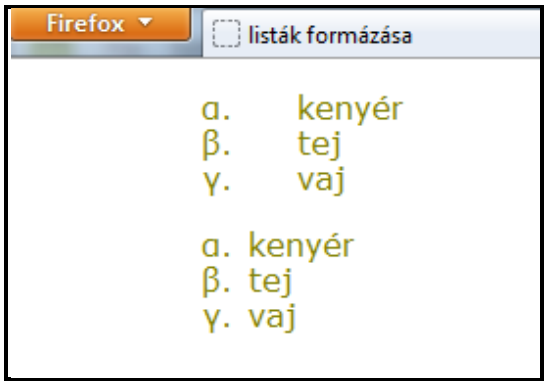
A böngészők így jelenítik meg a formázott listákat (alul a formázatlan, alapértelmezett állapot):



Megjegyzés: Az felsorolt lista-formázások mindig a teljes listára, mint egységes *<ol>*, *<ul>* vagy *<dl>* elemre vonatkoznak. Ezen felül még a lista soraihoz külön is rendelhetők további tulajdonságok, melyek definiálása a szokásos módon történik. Például ha a rendezett lista sorait beljebb akarjuk kezdeni az alapértelmezettnél, akkor a korábbi kódoláshoz még az alábbit kell hozzáírni:

```
ol li { padding-left:25px; }
```

Felül a <*li*> elemeket beljebb helyező, alul a <*li*> elemeket külön nem formázó megjelenítés:



(Egy adott sor vagy a soron belül egy adott elem megjelenítését pedig az *id* vagy *class* azonosításához rendelt tulajdonság és érték alapján lehet formázni.)

<u>A listák formázására felhasznált CSS tulajdonságok:</u>

list-style-type	lista stílusának típusa
list-style-image	felsorolásjel képe
list-style-position	felsorolásjel helyzete
list-style	összevont forma a lista tulajdonságok megadására

3.14. Standard vonaltípusú szögletes szegélyek

A szegélyt alkothatja előre meghatározott stílusú vonal vagy szabadon választott kép. Ebben a fejezetben az adott vonaltípusokból kialakítható szegélyek alkalmazása kerül bemutatásra, a képekből kialakított szegélyeket külön fejezet tárgyalja.

A szegély a háttér (ha van háttérszín vagy háttérkép megadva) elé, de a tartalom mögé (ha átfedés van a szegély és tartalom között) kerül.

Az előre meghatározott vonaltípusokból választható szegélyek esetében a szegély stílus (*border-style*), szegély szín (*border-color*) és szegély vastagság (*border-width*) tulajdonságok adhatóak meg különböző értékekkel.

A **szegély stílusok** lehetséges értékeit és megjelenítésüket a következő kép mutatja:

none (nincsen szegély) - ez az alapértelmezett érték

hidden (rejtett a szegély) - nem látszik a szegély

dotted (pontozott vonal) - a pontok alakja és egymástól való távolsága nem formázható

dashed (szaggatott vonal) - a szakaszok hossza és köztes távolsága nem formázható

solid (folytonos vonal)

double (dupla vonal) - a vonalak vastagsága és a közöttük lévő térköz egyenként nincsenek specifikálva, de a két vonalvastagság és a térköz összegének egyenlőnek kell lenniük a szegélyvastagsággal

groove (bemélyített vonal) - a szegély színéhez képest két, egyhén világosabb ill. sötétebb színnel „árnyékot" képez, ami bemélyedést utánoz

ridge (kidomborodó vonal)

inset (süllyesztett vonal)

outset (kiemelkedő vonal)

A szegélyek (*border*) kialakítása és elhelyezése az elemi doboz-modell alapján történik. A doboz négy szegélyének a stílusai külön-külön definiálhatók a *border-top-style* (felső szegély stílusa), *border-right-style* (jobb oldali szegély stílusa), *border-bottom-style* (alsó szegély stílusa), *border-left-style* (bal oldali szegély stílusa) tulajdonságokkal.

A weboldal fő címsorának szegélye (ha minden oldal más-más stílusban jelenik meg) pl. az alábbiak szerint kódolható:

```
h1 {
border-top-style:solid;
border-right-style:double;
border-bottom-style:dotted;
border-left-style:dashed; }
```

A fenti négy tulajdonság összevont (*shorthand*) formában *border-style* tulajdonságként is definiálható, az értékek megadási sorrendje kötelezően a tulajdonságok fenti sorrendjének (az óramutató járásával megegyező irány) betartásával történik. Tehát a fenti szegély stílus az alábbi módon is megadható:

```
h1 { border-style:solid double dotted dashed; }
```

Ha négynél kevesebb értéket adunk meg, a böngészők úgy értelmezik, hogy a hiányzó baloldali megegyezik a jobboldalival, és a hiányzó alsó érték megegyezik a felsőével. Ha valamennyi oldalszegélyhez ugyanaz az érték tartozik, elegendő az értéket egyszer megadni:

```
h1 { border-style:solid; }
```

A szegélyek **színei** a színekre vonatkozó (hexa, százalékos, tízes számrendszerű, színnév, hsl) értékmegadási lehetőségekkel definiálhatók.

A doboz négy szegélyének a színe külön-külön megadható a *border-top-color* (felső szegély színe), *border-right-color* (jobb oldali szegély színe), *border-bottom-color* (alsó szegély színe), *border-left-color* (bal oldali szegély színe) tulajdonságokkal.

Egy weboldal fő címsorának szegélye, ha minden oldal más-más színű, pl. az alábbiak szerint kódolható:

```
h1 {
border-top-color:#6cc;
border-right-color:red;
border-bottom-color:(0, 255, 255);
border-left-color:transparent; }
```

(A legutolsó nem fog látszódni, mivel átlátszó!)

A fenti négy tulajdonság összevont (*shorthand*) formában *border-color* tulajdonságként is definiálható, az értékek megadási sorrendje kötelezően a tulajdonságok fenti sorrendjének (az óramutató járásával megegyező irány) betartásával történik:

```
h1 { border-color:#6cc red (0, 255, 255) transparent; }
```

Ha négynél kevesebb értéket adunk meg, a böngészők úgy értelmezik, hogy a hiányzó baloldali megegyezik a jobboldalival, és a hiányzó alsó érték megegyezik a felsőével.

Ha valamennyi oldalszegélyhez ugyanaz a szín tartozik, elegendő az értéket egyszer megadni:

```
h1 { border-color:#6cc; }
```

A **szegély vastagsága** a *border-top-width* (felső szegély vastagsága), *border-right-width* (jobb oldali szegély vastagsága), *border-bottom-width* (alsó szegély vastagsága), *border-left-width* (bal szegély vastagsága) tulajdonsággal formázható.

Az érték megadható pixelben, százalékban (a befoglaló blokk szélessége a 100%), *em*-ben vagy *thin* (vékony), *medium* (közepes) – ez az alapértelmezett – és *thick* (vastag) kifejezésekkel (ezek vastagsága pontosan nem definiált, az egyes böngészők alapbeállítása szabja meg, de egy dokumentumon belül állandó érték):

```
h1 {
border-top-width:10px;
border-right-width:1em;
border-bottom-width:5%;
border-left-width:medium; }
```

A mértékegység üres betűköz nélkül követi az értéket!

A százalékos értéket önmagában nem mindig értelmezik a böngészők – kell egy abszolút szám, ami viszonyítási alapként szolgál.

Figyelem! Bár a szegély vastagsága alapértelmezetten *medium*, ugyanakkor az alapértelmezett szegély stílus a *none*, tehát ha mindkét tulajdonság értéke alapértelmezett, akkor 0 vastagságú szegély lesz, azaz *medium* helyett nem lesz szegély.

A *border-width* összevont tulajdonság alkalmazásával a *top, right, bottom* és *left* értékek a fenti kötött sorrendben együttesen megadhatók:

```
h1 { border-width:3px medium 1em 5%; }
```

Ha négynél kevesebb értéket adunk meg, a böngészők úgy értelmezik, hogy a hiányzó baloldali megegyezik a jobboldalival, és a hiányzó alsó érték megegyezik a felsőével.

Ha valamennyi oldalszegélyhez ugyanaz a vastagság tartozik, elegendő az értéket egyszer megadni:

```
h1 { border-width:15px; }
```

Ugyancsak összevonási lehetőség a *border-top*, *border-right*, *border-bottom*, *border-left* tulajdonságok használata a *border-width*, *border-style*, *border-color* (kötött) sorrendben:

```
h1 {
border-top:10px solid #66cccc;
border-right:1em double red;
border-bottom:5% dotted (0, 255, 255);
border-left:medium dashed transparent; }
```

A szegély stílus, szín és vastagság a *border* tulajdonsággal még tovább összevonható, azonban ez az összevonás (*shorthand*) csak akkor használható, ha a négy szegély stílusa, színe és vastagság értékei is egyenlőek:

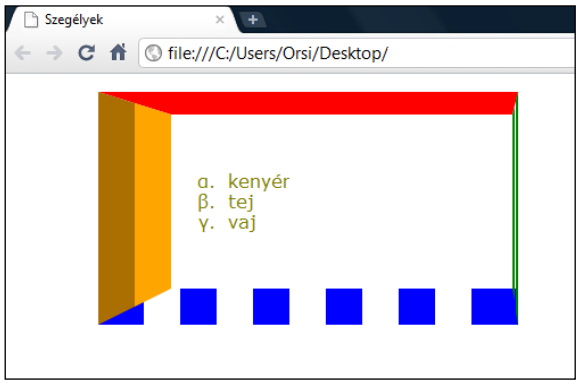
```
h1 { border:solid green 15px; }
```

Ha szegélyenként eltérő érték fordul elő, a korábbi, részletező (*longhand*) tulajdonságokat kell alkalmazni.

Példaként az előző fejezetben formázott élelmiszeres rendezett listát bekeretezzük úgy, hogy szegélyének minden oldala más-más színű, stílusú és vastagságú legyen:

```
ol { color:olive; font-family:Verdana, Geneva, sans-serif;
   list-style:lower-greek outside; margin-left:75px; border-
   style:solid double dotted groove; border-color:red green
   blue orange; border-width:20px thick 2em 4em; padding:50px;
   width:200px; }
```

A megjelenítés az alábbi lesz:



A kódolással és megjelenítéssel kapcsolatos megjegyzések:

- a listaelemnek azért adtunk szélességet (200px), hogy ne a teljes vízszintesen szabad területet keretezzék be a böngészők
- az eltérő színek találkozásánál a színátmenet a szegélyvastagságok hányadosából eredő meredekség mentén következik be
- egyes eltérő stílusú szegélyvonalak találkozásuknál nem tudnak illeszkedni egymáshoz – lásd bal alsó sarok)

Figyelem! Ha bármilyen összevont (*shorthand*) formában történik kódolási hiba, az valamennyi értéket érvényteleníti, nem csak a hibásan kódoltat. Részletező (*longhand*) kódolásnál viszont csak az érvénytelenül kódolt tulajdonság/értéknek nem érvényesül a hatása.

A standard vonaltípusú szegélyekkel **néhány speciális megoldás** is kialakítható:

a) szöveg aláhúzása (pl. amíg az új *text-decoration* tulajdonságokat nem értelmezik a böngészők) – minden szót egymástól és a szöveg színétől eltérően húzhatunk alá:

HTML:

```
<p>
<span id="elso">minden</span><span id="masodik"> szín</span>
<span id="harmadik"> különböző</span>
</p>
```

CSS:

```
#elso { color:gray; font-size:xx-large; border-top:0px solid
aqua; border-right:0px solid aqua; border-bottom:4px solid
aqua; border-left:0px solid aqua; padding-bottom:1em; }
#masodik { color:green; font-size:xx-large; border-top:0px
dashed red; border-right:0px dashed red; border-bottom:3px
dashed red; border-left:0px dashed red; }
#harmadik { color:yellow; font-size:xx-large; border-top:0px
dotted fuchsia; border-right:0px dotted fuchsia; border-
bottom:2px dotted fuchsia; border-left:0px dotted fuchsia;
padding-bottom:0.5em; }
```

Szándékosan nem használtunk összevont alakokat a kódolásban, hogy tételesen látszódjon, a szegélyek közül az alsó kivételével valamennyit eltűntettük, így lett belőlük aláhúzás:

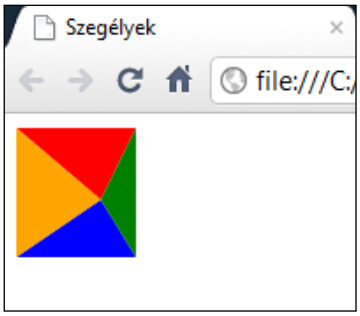
b) háromszög/nyíl kialakítása

HTML:

```
<div></div>          ( üres, nulla méretű terület )
```

CSS:

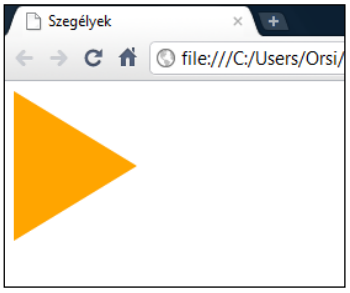
```
div { width:0; border-top:40px solid red; border-right:20px
solid green; border-bottom:32px solid blue; border-left:48px
solid orange; }
```



Egy üres, 0-méretű *div* köré szegélyeket rakva egyrészt a fentihez hasonló díszítő-elemek, másrészt – elhagyva/eltakarva a 4 háromszög közül hármat – nyilak/háromszögek alakíthatók ki:

```
div { border-top: 60px solid transparent;
      border-bottom: 60px solid transparent;
      border-left: 100px solid orange;
      width:0; }
```

Valójában 2 oldalt láthatatlanná (átlátszóvá) tettük a szegélyeket, és egy (a jobb) oldalon nem adtunk meg vastagságot, tehát ott nem láthatatlan, hanem nincsen szegély. Ha a két láthatatlan szegélyt nem kódoltuk volna, akkor a negyedik, kódolt szegély sem tudna kialakulni:

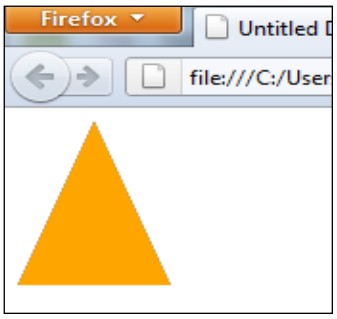


Az előző eset analógiájára a balra mutató nyíl CSS-kódja:

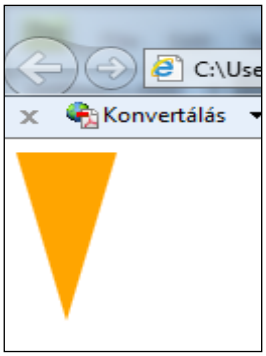
```
div { width: 0px;
border-top: 60px solid transparent;
border-bottom: 60px solid transparent;
border-right: 100px solid blue; }
```

Felfelé/lefelé mutató, és eltérő méretarányú nyilak/háromszögek ugyancsak a fentiek szerint kódolhatók:

```
div { width: 0px;
border-left: 40px solid transparent;
border-bottom: 100px solid orange;
border-right: 25px solid transparent; }
```



```
div { width: 0px;
border-left: 25px solid transparent;
border-right: 40px solid transparent;
border-top: 100px solid orange; }
```

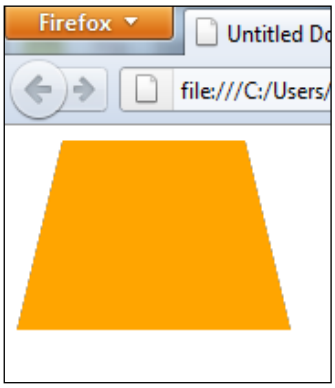


c) trapéz kialakítása

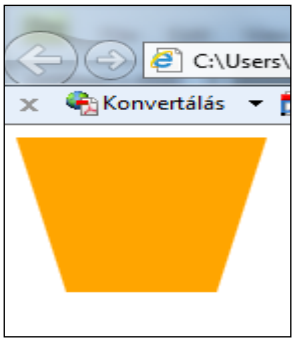
Ha a szegéllyel ellátott *div* szélessége nem 0, akkor háromszög/nyíl helyett trapéz adódik, melynek a rövidebb oldala a *div* szélessége, hosszabb oldala a látható szegély vastagsága.

A fenti egyenlő szárú háromszögek/nyilak egyenlő szárú trapézokká válnak, ha a *div* szélességének véges pozitív értéket adunk, pl.:

```
div { width:100px;
border-left:25px solid transparent;
border-right:25px solid transparent;
border-bottom:100px solid orange; }
```




```
div { width:75px;
border-left:25px solid transparent;
border-right:25px solid transparent;
border-top:100px solid orange; }
```



Figyelem! A szegélyekből kialakított fenti alakzatokba szöveg vagy egyéb tartalom nem helyezhető be és (további) szegéllyel nem láthatóak el.

d) levágott szegélysarkok

Levágott szegélysarkok előállíthatóak olyan módon, hogy a lehető legkisebb (*1px*) magasságú, változó szélességű üres *div*-eket illesztünk alul-felül az eredeti szögletes dobozhoz:

Eltüntetve a csatlakozó felületeknél lévő nemkívánatos szegélyeket és a dobozzal azonos hátteret adva, a hozzáadott *div*-ek pontszerű oldalszegélyei alkotják a levágott sarok kontúrját.

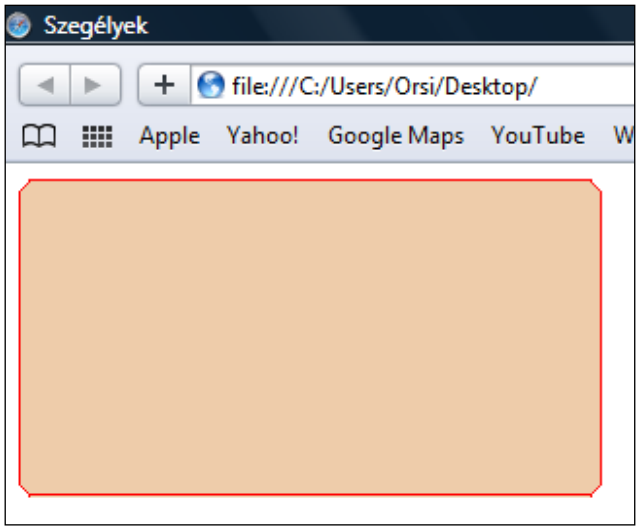
45°-os ferdeségű levágás (valójában hozzátoldás) esetén, 5-5 *div* hozzáadását kódolva, melyek szélessége alul-felül 1-1 px-el folyamatosan változik (csökken):

```
<style>
  .doboz { width:300px; }
    .doboz .x1, .doboz .x2, .doboz .x3, .doboz .x4, .doboz .x5
     { height:1px; background:#eca; border-left:1px solid red;
       border-right:1px solid red; }
    .doboz .x1 { margin:0px 5px; border-top:1px solid red;
       height:0px; }
    .doboz .x2 { margin:0px 4px; }
    .doboz .x3 { margin:0px 3px; }
    .doboz .x4 { margin:0px 2px; }
    .doboz .x5 { margin:0px 1px; }
  .doboz .tartalom { background:#eca; border-left:1px solid
                red; border-right:1px solid red; height:150px; }
</style>
```

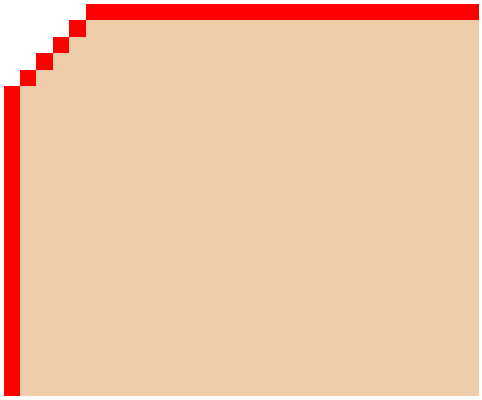
HTML:

```
<div class="doboz">
  <div class="x1"></div><div class="x2"></div><div class="x3">
  </div><div class="x4"></div><div class="x5"></div>
  <div class="tartalom"></div>
  <div class="x5"></div><div class="x4"></div><div class="x3">
  </div><div class="x2"></div><div class="x1"></div>
</div>
```

A vízszintes szegélyeket adó, *x1* azonosítójú *div* magasságát *0*-ra vettük vissza, hogy csak a szegélyt adja, és ne okozzon törést a pontokból összeálló ferde egyenesen. Miután kódja közelebb van a tartalomhoz mint a valamennyi *x* –re vonatkozó *1 px* közös kódolás, felülírja azt.



Figyelem! Értelemszerűen nagyméretű ábráknál ez a megoldás már nem használható, mert látszódni fognak az egyes képpontok a folytonos vonal helyett.



Megjegyzés: A (tetszőleges szögben) levágott sarkok kódolással történő megvalósítását a CSS4 vonatkozó moduljában tervezik bevezetni.

<u>A standard vonaltípusú szögletes szegélyek formázására felhasznált tulajdonságok:</u>

border-style: border-top-style, border-right-style, border-bottom-style, border-left-style

border-color: border-top-color, border-right-color, border-bottom-color, border-left-color

border-width: border-top-width, border-right-width, border-bottom-width, border-left-width

border-top: border-top-width, border-top-style, border-top-color

border-right: border-right-width, border-right-style, border-right-color

border-bottom: border-bottom-width, border-bottom-style, border-bottom-color

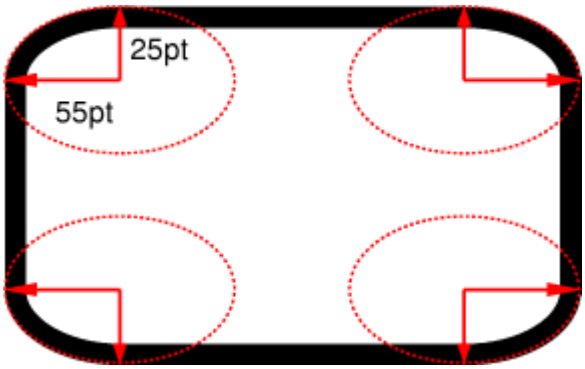
border-left: border-left-width, border-left-style, border-left-color

border: border-style, border-color, border-width

3.15. Standard vonaltípusú lekerekített szegélyek

A **szegélyek lekerekítése** a *border-top-left-radius* (bal felső sarok sugara), *border-top-right-radius* (jobb felső sarok sugara), *border-bottom-right-radius* (jobb alsó sarok sugara), *border-bottom-left-radius* (bal alsó sarok sugara) tulajdonsággal formázható.

A lekerekítés negyed-ellipszisek formájában történik, a sarkonként megadott 2-2 sugár a külső szegélyél alakját határozza meg.



Az egyes sarkoknak nem szükségszerűen kell azonos lekerekítéssel rendelkezniük, eltérő lekerekítésekkel aszimmetrikus alakzatok is kialakíthatók.

A lekerekítési sugarak *pt*-ben, *px*-ben, %-ban vagy *em*-ben egyaránt megadhatók, a két érték közül az első mindig a vízszintes sugarat, a második a függőleges sugarat jelöli. Ha a második sugár értéke nincsen megadva, akkor az egyenlőnek tekintendő az elsőével (egy saroknál azonos sugarak esetén az ellipszis speciális eseteként negyedkört kapunk). Ha bármelyikük értéke 0, akkor szögletes (nem lekerekített) az adott szegélysarok. Százalékos értékmegadás esetén a vízszintes sugárnál a szegélydoboz szélessége, a függőleges sugárnál a szegélydoboz magassága a 100%.

Egy eltérően lekerekített sarkokkal rendelkező szegély kódolása pl.:

```
border-top-left-radius: 55pt 25pt;
border-top-right-radius: 50px 30px;
border-bottom-right-radius: 25% 40%;
border-bottom-left-radius: 10em 5em;
```

A fenti lekerekítési sugarak összevont (*shorthand*) alakban megadott kódolással:

```
border-radius: 55pt 50px 25% 10em / 25pt 30px 40% 5em;
```

A / („per”) jel előtt a vízszintes sugarak, utána a függőleges sugarak szerepelnek. A négy sarokra vonatkozó értékek a bal felső sarokból indulva az óramutató járási sorrendjében (bal felső, jobb felső, jobb alsó, bal alsó) vannak megadva.

Ha a bal alsó érték nincs megadva, akkor egyenlő a jobb felsővel. Ha a jobb alsó nincs megadva, akkor egyenlő a bal felsővel. Ha a jobb felső nincsen megadva, akkor egyenlő a bal felsővel. Ha nincsen / jel, az értékek mindkét irányú sugárra vonatkoznak.

Fentiek értelmében ha valamennyi sarok azonos negyed-ellipszis lekerekítésű, és a vízszintes féltengelyek pl. *20px*, a függőlegesek pl. *30px* méretűek, akkor összevontan a kódolás:

```
border-radius: 20px / 30px;
```

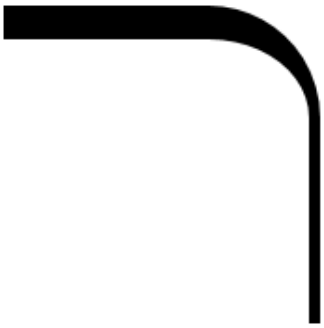
Ha pedig valamennyi sarok azonos negyedkör lekerekítésű, és a sugár pl. 20%, akkor összevontan a kódolás:

```
border-radius: 20%;
```

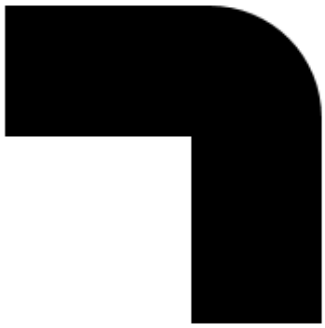
Valamennyi szegély-stílus (pontozott, szaggatott, süllyesztett, stb.) követi a szegély lekerekítését.

Ha nem azonos vastagságú szegélyek találkoznak egy lekerekített saroknál, a lekerekítés során folyamatos átmenet lesz az egyik vastagságból a másikba.

Ha a szegély vastagsága nagyobb, mint a lekerekítési sugár, csak a szegély külső éle lesz lekerekítve, a belső szegély szögletes marad:

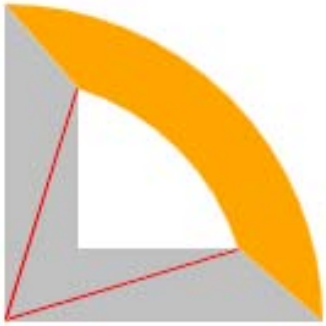


szegélyvastagság folyamatos átmenete



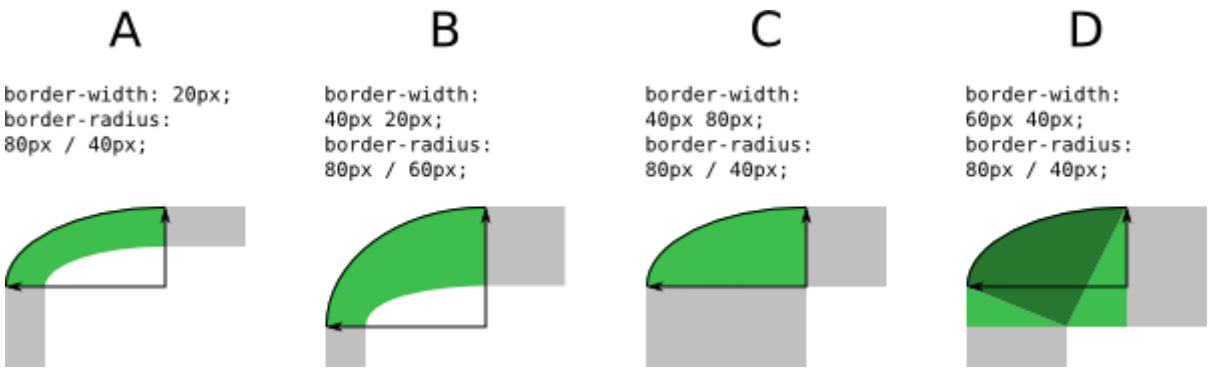
szegély vastagabb a lekerekítési sugárnál

Ha a lekerekítés sugara nagyobb, mint a szemközti saroktól mért távolság, akkor a lekerekítés íve 90°-osnál kisebb lesz:



A sarok lekerekítéseknek nem szabad átfedniük egymást, azaz két szomszédos szegélysarok sugarának összege nem lehet nagyobb a szegélydoboz méreténél. Ha mégis meghaladják ezt a méretet, a böngészők arányosan lecsökkentik a sugarak méretét az átfedés határhelyzetéig. Ez a megjelenítésben a kódoláshoz képest kisebb eltérést okoz.

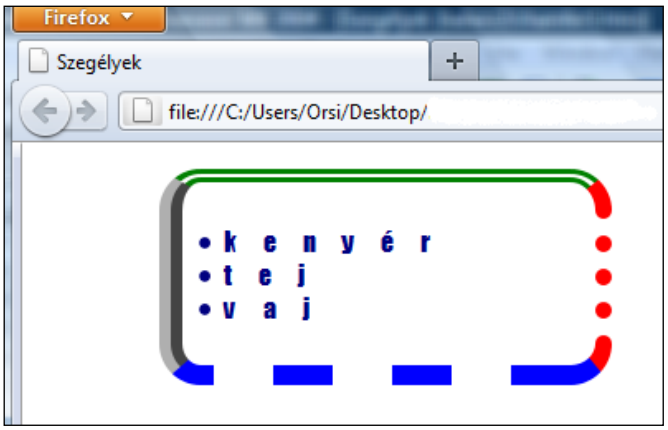
Az érintkező szegélyek stílus- és színátmenetének is meg kell történnie a lekerekítés során. Az átmenet középpontja a lekerekítési görbe azon pontja, melynek vízszinteshez viszonyított szöge arányos a szegélyvastagságok hányadosával. Pl. ha a felső és jobb szegély vastagsága egyenlő (A. ábra), akkor a hányados 1, tehát 45°-os szögnél lesz az átmenet. Ha a felső szegély kétszer vastagabb a jobb oldalinál (B. ábra), az átmenet a vízszinteshez képest 30°-ra lesz, stb. Ennek a szögnek mentén, a szegély külső és belső ívét összekötő egyenesen lesz az éles szín- és stílusátmenet (folyamatos szín- és stílusátmenet nem kódolható).



A fenti lekerekítési adatoknál az átmenetek a zölddel jelölt területekre esnek. A D. ábra esetében azonban a lekerekítési sugarak által definiált négyszög nem foglalja magába a belső görbe középpontját (szögletes sarok), így az átmeneti területet ki kell úgy terjeszteni (sötétzöld terület), hogy a szögletes sarok is belekerüljön.

Példaként az előző fejezetben formázott élelmiszeres felsorolási listát bekeretezzük úgy, hogy szegélyének minden oldala más színű, stílusú és vastagságú vonal legyen. A CSS kódolásban a szegélyek tulajdonságai/értékei pirossal vannak jelölve:

```
ul { color:navy; letter-spacing:1em; font-family:Impact,
     Charcoal, sans-serif; list-style:square inside; border-
     width:8px 10px 12px 14px; border-style:double dotted
     dashed ridge; border-color:green red blue black; border-
     radius:25px; padding:25px; width:200px; margin-left:75px;
}
```



Figyelem! Internet Explorer 8 nem értelmezi a *border-radius* tulajdonságot, szögletes sarokkal jeleníti meg a fenti kódolást.

Lekerekített szegélyekkel **néhány speciális alakzat**ot is ki lehet alakítani:

a) Nulla szegélyvastagságú, háttérszínnel rendelkező *div* esetén:

- pötty, korong

Ha egy négyzet alakú *div* (*width = height*) szegélyének a lekerekítési sugara a fele az elem szélességének (és magasságának), akkor a háttérszínének megfelelő korongot kapunk:

```
div { width:.................px; height:..............px; border-radius:50%;
      background-color:..............; }
```

- szabályos ovális alakzat

Ha a *div* téglalap alakú (*width ≠ height*), és a lekerekítési sugarak a magasság ill. hosszúság felével egyenlőek, akkor szabályos, vízszintes vagy függőleges ovális alakzatot kapunk:

```
div { width:…….px; height:……..px; border-radius:50%;
      background-color:……; }
```

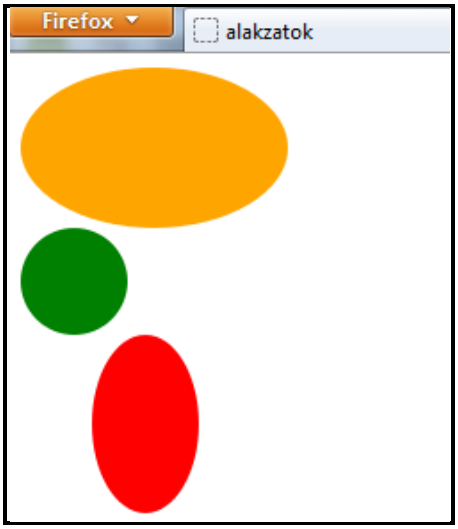
A fenti alakzatok kódolása és megjelenítése:

HTML:

```
<div id="ovalisegy"></div>
<div id="korong"></div>
<div id="ovalisketto"></div>
```

CSS:

```
#ovalisegy { width:150px; height:90px; border-radius:50%;
             background-color:orange; }
#korong { width:60px; height:60px; border-radius:50%;
          background-color:green; }
#ovalisketto { width:60px; height:100px; border-radius:50%;
          background-color:red; margin-left:40px; }
```



Az alakzatokba tartalom is bevihető, pl. egyszerű szöveg esetén:

HTML:

```
<div id="ovalisegy">valami</div>
<div id="korong">más valami</div>
```

CSS:

```
#ovalisegy { width:150px; height:90px; border-radius:50%;
             background-color:orange; text-align:center; font-
             size: 28px; color:#00F; }
#korong { width:60px; height:60px; border-radius:50%;
          background-color:green; text-align:center; font-
          size: 18px; color:#FF0; }
```

- szabálytalan (amorf) gömbölyű formákkal kialakított alakzatok

Ha egy blokkszintű elem valamennyi sarkát mind vízszintes mind függőleges irányban eltérő sugarakkal kerekítjük le és 0 szegélyvastagságot állítunk be, szabálytalan legömbölyített (amorf) alakzat alakítható ki, mely nemcsak díszítőelemként szolgálhat, hanem tartalmat is el lehet benne helyezni.

Példaként a négy kezdő címsorunkat egy „amorf” azonosítójú *div* szakaszba rakjuk és sarkait lekerekítve, háttérszínnel ellátva formázzuk:

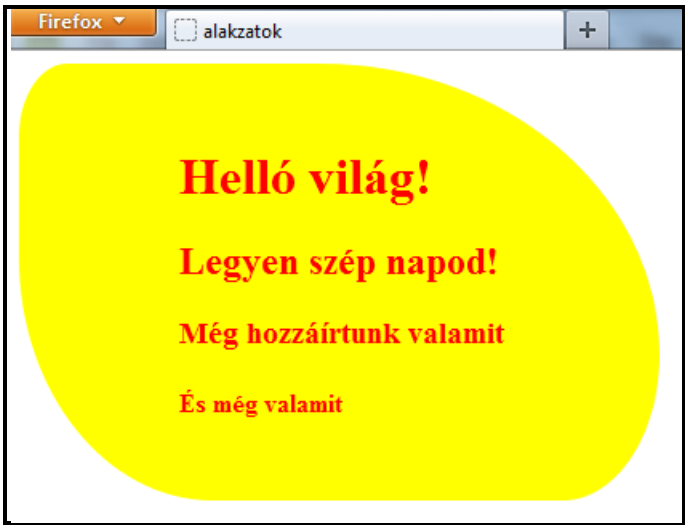
HTML:

```
<div id="amorf">
     <h1>Helló világ!</h1>
     <h2>Legyen szép napod!</h2>
     <h3>Még hozzáírtunk valamit</h3>
     <h4>És még valamit</h4>
</div>
```

CSS:

```
#amorf {
   width:300px;
   background-color:yellow;
   color:red;
   padding-left:100px;
   padding-top:30px;
   padding-bottom:30px;
   border-top-left-radius:2em 3em;
   border-top-right-radius:14em 12em;
   border-bottom-right-radius:4em 6em;
   border-bottom-left-radius:8em 10em;
}
```

A nem egyformán lekerekített sarkú (amorf), nulla szegélyvastagságú, kontúros, színes háttérrel kialakított elem:



b) gombok

A (nyomó)gombok kódolással való kialakítása már a standard vonaltípusú szögletes szegélyekkel is megvalósítható. A **<button>**……**</button>** HTML-kódhoz (a valóságban valamilyen utasítást tartalmaz a gomb, de most csak a formázás az érdekes, úgyhogy nem foglalkozunk a funkciójával) egy CSS kódolás:

```
button { width:160px; height:40px; text-align:center;
     color:blue; font-family:Corsiva, serif; font-weight:bold;
     font-size:24px; background-color:yellow; }
```

A <*button*>-hoz alapértelmezetten hozzárendelt félszegély a *border* tulajdonság/értékkel tovább formázható, pl.:

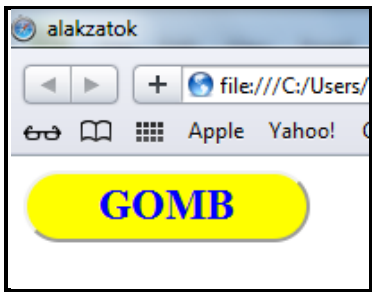
```
button { width:160px; height:40px; text-align:center;
     color:blue; font-family:Corsiva, serif; font-weight:bold;
     font-size:24px; border:2px solid green; background-color:
     yellow; }
```

Megfelelő szegély stílussal (*groove/ridge* vagy *inset/outset*) akár a benyomott/normál állapot is imitálható.

Esztétikusabb és divatosabb azonban a lekerekített sarkú, szegély nélküli gombok használata. A lekerekített sarkok a *border-radius* tulajdonsággal (a benyomott/normál állapot imitálása pedig a később ismertetett doboz-árnyék és/vagy színátmenet tulajdonságokkal) kódolható.

Egy egyszerű lekerekített gomb megvalósítása:

```
button { width:160px; height:40px; text-align:center; font-
family:Corsiva, serif; color:blue; font-weight:bold;
background-color:yellow; border-radius:20px; font-size: 24px;
}
```



A <*button*>-hoz alapértelmezetten hozzárendelt félszegély a *border:none;* tulajdonság/ értékkel tüntethető el:

```
button { width:160px; height:40px; text-align:center; font-
         family:Corsiva, serif; color:blue; font-weight:bold;
         background-color:yellow; border-radius:20px; font-
         size: 24px; border: none; }
```

Figyelem! A *border-radius* tulajdonság nem vonható össze a *border-style*, *border-color* és *border-width* tulajdonságokkal a *border* rövid alak égisze alatt, mert új, és nem illeszkedik a már korábban kialakított *shorthand* sémához.

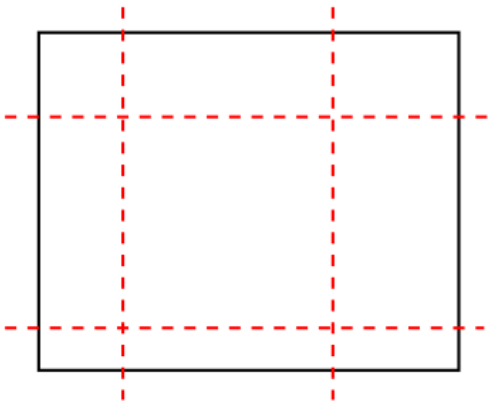
<u>A standard vonaltípusú lekerekített szegélyek formázására felhasznált tulajdonságok:</u>

border-radius: border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius

3.16. Képből készített szegélyek

Egy kötött algoritmus segítségével egy tetszőleges képből is készíthető szegély. A – célszerűen a gyűjtőmappánkban tárolt – képet két vízszintes és két függőleges vonallal 9 da-

rabra osztjuk, és ezek a darabok alkotják majd a szegély oldalait, sarkait, ill. a középső rész kihagyásával vagy megtartásával a tartalom doboz és belső margó hátterét.



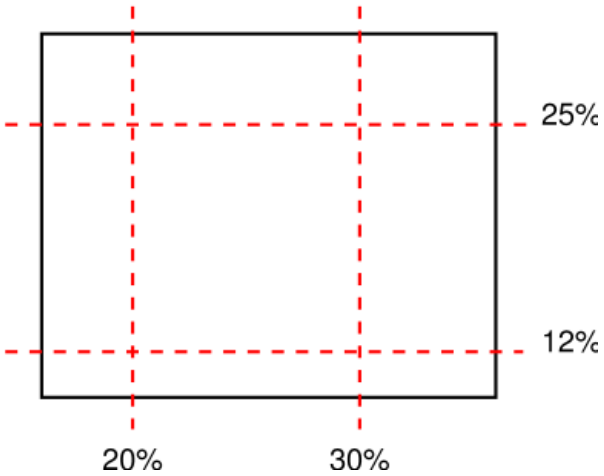
Annyiban kötött az algoritmus, hogy a szegély kialakítása csak egy képből (nem pedig esetleg több kép kombinációjából) hozható létre, a kép csak a leírt módon darabolható (ferde egyenesek vagy kilencnél több rész nem lehetséges), és a szegélyrészek helyzete csak a kép darabolási sémája szerinti lehet (nem cserélhetők fel oldalak).

Egy képből készített szegély a *border-image* tulajdonsággal hozható létre, mely megadható összevont (*shorthand*) és részletezett (*longhand*) alakban. Az ismertetésnél a jobb áttekinthetőség érdekében a részletezett formát használjuk.

A szegély kialakításához felhasznált képet a *border-image-source* tulajdonsággal definiáljuk, melynek értéke vagy *none* – ekkor nem lesz képből szegély – vagy a kép elérési útja. Ha a képet a HTML/CSS dokumentummal közös gyűjtőmappában tároljuk, forrásként elég a fájlnév és kiterjesztés megadása:

```
border-image-source:url(fájlnév.kiterjesztés);
```

A *border-image-slice* tulajdonság a kép szélétől mérve a teljes képméret (100%) százalékában adja meg, hogy mekkora részeket használunk fel a képből a szegély céljára. Pl.:



Az értékeket szóközzel, vessző nélkül felsorolva kell megadni, a kötelező sorrend fentről indulva az óramutató járásával megegyező irány (a kép felső, jobb oldali, alsó, és bal oldali élei), például a fenti ábra szerint:

```
border-image-slice:25% 30% 12% 20%;
```

Ha a bal oldali él értéke hiányzik, akkor az megegyezik a jobb oldalival, ha az alsó hiányzik, akkor az megegyezik a felsővel, ha a jobb oldali hiányzik, akkor az azonos a felsővel.

Negatív értékek nem értelmezettek, a 100%-nál nagyobb értékeket (a képnél nagyobb képdarabokat) 100%-nak veszi a szabvány.

A kép középső része alapértelmezetten üresnek tekintett. Ha a *fill* kulcsszót is definiáljuk a százalékos értékek után, akkor látható marad, és ha van háttér, akkor azt elfedi.

Az egymással szemben lévő kijelölt képdarabok átfedhetik egymást. Ha a jobb és bal oldali élek szélességének összege egyenlő vagy nagyobb mint a kép szélessége, akkor a szegély alsó és felső része, továbbá a középső rész üres lesz. Analóg módon, ha a felső és alsó élek magasságának összege egyenlő vagy nagyobb mint a kép magassága, akkor a szegély jobb és bal oldali része, továbbá a középső rész lesz üres.

A szegély oldalainak szélességét a *border-image-width* tulajdonság relatív (*%*) vagy abszolút (*px*) nagyságban megadott értékei definiálják. Az értékek sorrendje itt is felső, jobb oldali, alsó és bal oldali szegély. Pl.:

```
border-image-width:40px 30px 20px 30px;
```

Ha a bal oldali érték hiányzik, akkor az megegyezik a jobbal, ha az alsó hiányzik, akkor az azonos a felsővel, ha a jobb hiányzik, akkor az megegyezik a felsővel.

Negatív értékek nem értelmezettek. Ha két szemben lévő szegély él a megadott nagy szélességeik miatt átfedné egymást, a szabvány arányosan lecsökkenti a szélességeket az átfedés határhelyzetéig.

További lehetséges érték az *auto*, ekkor a szegély kialakításához felhasznált kép kijelölt darabjának eredeti, saját (*intrinsic*) mérete lesz a szegély szélessége.

A *border-image-outset* tulajdonsággal a szegély helyzete a szegély doboztól kifelé tolható. Négy értéke az ismert módon a felső, jobb, alsó és bal oldalra vonatkozik. Ha hiányzik a bal oldali, akkor az megegyezik a jobbal, ha az alsó hiányzik, akkor az azonos a felsővel, ha a jobb oldali hiányzik, akkor az egyenlő a felsővel:

```
border-image-outset:20px;
```

A *border-image-repeat* tulajdonság a képdarabnak a szegélyben történő ismétlődését szabályozza, lehetséges értékei:

-*stretch:* ez az alapértelmezett, a képdarab olyan mértékben van széthúzva, hogy kitöltse a szegélyszakaszt

- *repeat*: a teljes kitöltendő szegélyszakaszt a képdarab méretének változatlanul tartásával, annak ismétlésével összefüggően kitapétázza

- *round:* ha a képdarab egésszámú többszöröse nem pontosan tölti ki a szegélyszakaszt, akkor úgy méreteződik át, hogy egésszámú többszöröse illeszkedjen a szegélyszakasz kitöltéshez

- *space:* az első és utolsó képdarab a szegélyszakasz széleihez illeszkedik, és a többi képdarab változatlan méretben olyan térközzel helyezkedik el közöttük, hogy egyenletesen kitöltsék a szegélyszakaszt

Két kulcsszóval történik az értékek megadása, az első a vízszintes, a második a függőleges szegélyszakaszokra vonatkozik. Pl.:

```
border-image-repeat:round space;
```

Ha a második érték nincsen megadva, azonosnak tekintendő az elsővel.

A fentiek alapján *200px* x *100px*-es *div*-eket fogunk az alábbi képből készült különféle szegélyekkel ellátni:

a) Két *div*-et helyeztünk egymás mellé, és mindkettőbe beírtuk, hogy mit kódoltunk a szegélyére vonatkozóan. Az eltérés a szegélyt alkotó képdarabok ismétlődési módjában van, egyébként mindkét esetben oldalanként azonos (20%-20%) arányban választottunk képdarabot, melyekből oldalanként különböző *(40px 30px 20px 30px)* szegélyszélességeket alakítottunk ki.

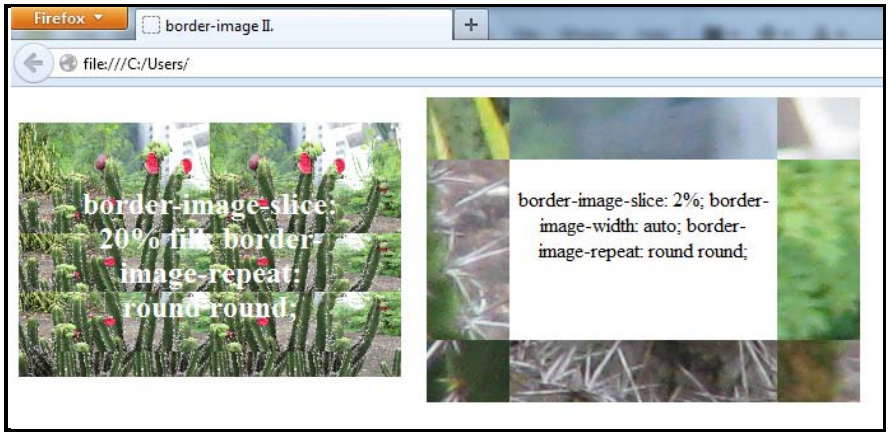
```
<!doctype html>
<html lang="hu">
<head>
	<title>border-image I.</title>
	<style>
		div {
			width:200px;
			height:100px;
			padding:50px;
			text-align:center; }
		#round {
			border-image-source:url(IMG_0945.jpg);
			border-image-slice:20% 20% 20% 20%;
			border-image-width:40px 30px 20px 30px;
			border-image-outset:0px;
			border-image-repeat:round round; }
		#stretch {
			border-image-source:url(IMG_0945.jpg);
			border-image-slice:20% 20% 20% 20%;
			border-image-width:40px 30px 20px 30px;
			border-image-outset:0px;
			border-image-repeat:stretch stretch;
			position:relative;
			top:-200px; right:-320px; }
	</style>
</head>
<body>
	<div id="round">border-image-slice: 20%; border-image-
		repeat: round round;</div>
	<div id="stretch">border-image-slice: 20%; border-image-
		repeat: stretch stretch;</div>
</body>
</html>
```

A megjelenítés:

b) Két *div*-et helyeztünk egymás mellé, és mindkettőbe beírtuk, hogy mit kódoltunk a szegélyére vonatkozóan. A bal oldali *div*-ben az előző esethez képesti eltérés a *fill* érték alkalmazása, melynek következtében háttérként jelenik meg a középső képdarab. A jobb oldali *div* szegélyét *20px*-el távolabbra toltuk, a szegély szélessége pedig *auto* értékre lett állítva. Ennek következtében a kép eredeti mérete érvényesül, és tekintettel annak nagy saját méretére, csak 2%-os darabot használhatunk szegélynek (a korábbi 20%-al szemben).

```
<!doctype html>
<html lang="hu">
  <head>
     <title>border-image II.</title>
     <style>
          div {
               width:200px;
               height:100px;
               padding:50px;
               text-align:center; }
          #fill {
               border-image-source:url(IMG_0945.jpg);
               border-image-slice:20% 20% 20% 20% fill;
               border-image-width:40px 30px 20px 30px;
               border-image-outset:0px;
               border-image-repeat:round round;
               position:relative;
               top:20px;
               color:#FFF;
               font-size:x-large;
              font-weight:bold; }
          #auto {
               border-image-source:url(IMG_0945.jpg);
               border-image-slice:2% 2% 2% 2%;
               border-image-width:auto;
               border-image-outset:20px;
               border-image-repeat:repeat repeat;
               position:relative;
               top:-180px; right:-340px; }
     </style>
  </head>
  <body>
     <div id="fill">border-image-slice: 20% fill; border-
          image-repeat: round round;</div>
     <div id="auto">border-image-slice: 2%; border-image-width:
          auto; border-image-repeat: round round;</div>
  </body>
</html>
```

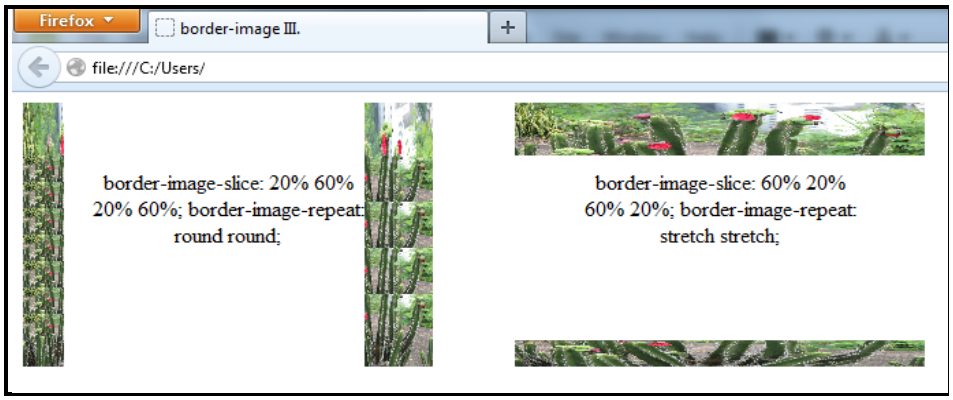
A megjelenítés:



c) Két *div*-et helyeztünk egymás mellé, és mindkettőbe beírtuk, hogy mit kódoltunk a szegélyére vonatkozóan. A bal oldali *div*-ben a jobb és bal oldali képdarabok méretének összege meghaladja a 100%-ot, ezért alul és felül nem lesz szegély. A jobb és bal oldali szegélyszakaszokat eltérő szélességűre kódoltuk. A jobb oldali *div* –ben a felső és alsó képdarabok méretének összege haladja meg a 100%-ot, ezért jobb és bal oldalt nem lesz szegély. A jobb és bal oldali szegélyszakaszokat itt is eltérő szélességűre kódoltuk.

```
<!doctype html>
<html lang="hu">
  <head>
     <title>border-image III.</title>
     <style>
          div {
               width:200px;
               height:100px;
               padding:50px;
               text-align:center; }
          #atfedes-vizsz {
               border-image-source:url(IMG_0945.jpg);
               border-image-slice:20% 60% 20% 60%;
               border-image-width:40px 50px 20px 30px;
               border-image-outset:0px;
               border-image-repeat:round round; }
          #atfedes-fugg {
               border-image-source:url(IMG_0945.jpg);
               border-image-slice:60% 20% 60% 20%;
               border-image-width:40px 50px 20px 30px;
               border-image-outset:0px;
               border-image-repeat:stretch stretch;
               position:relative;
               top:-200px; right:-360px; }
     </style>
  </head>
  <body>
     <div id="atfedes-vizsz">border-image-slice: 20% 60% 20% 60%;
          border-image-repeat:round round;</div>
     <div id="atfedes-fugg">border-image-slice: 60% 20% 60% 20%;
          border-image-repeat:stretch stretch;</div>
  </body>
</html>
```

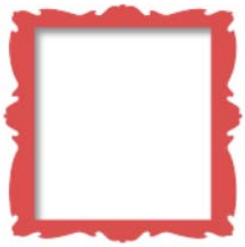
A megjelenítés:



A *border-image* tulajdonságok értékei összevontan is megadhatók, ekkor az értékek kötelező sorrendje: *border-image-source, border-image-slice, border-image-width, border-image-outset, border-image-repeat.*

Figyelem! A *border-image* tulajdonságot az Internet Explorer 11 előtti IE-verziók nem értelmezik.

A fenti kép pusztán demonstrációt szolgált, a gyakorlatban keresni vagy készíteni kell igazi szegélynek való képet. Például az alábbi képet felhasználva egy *div* szegélyének:



A formázás kódolása:

```
<style>
    div { width:300px;
          height:200px;
          background-color:yellow;
          text-align:center;
          padding: 30px;
          border-image-source:url(kepkeret.png);
          border-image-slice:20%;
          border-image-width:15%;
          border-image-outset:0px;
          border-image-repeat:stretch; }
</style>
```

A képből készített szegélyek formázására felhasznált tulajdonságok:

border-image-source:	none, url
border-image-slice:	méretek, fill
border-image-width:	méretek, auto
border-image-outset:	méretek
border-image-repeat:	stretch, repeat, round, space
border-image:	border-image-source, border-image-slice, border-image-width, border-image-outset, border-image-repeat

3.17. Körvonal

Az *outline* (körvonal, kontúr) tulajdonság valamilyen elem (pl. gomb, képtérkép, stb.) vizuális kihangsúlyozására szolgál. Mind kódolásában, mind megjelenítésben hasonlít a standard vonaltípusú szegélyekre, de vannak lényeges eltérések:

- csak szögletesek lehetnek
- nem foglalnak el helyet
- tetszőleges alakú elemhez négyzetként illeszkednek
- a körvonal egyes oldalait nem lehet külön-külön formázni

A tulajdonságok részletezve *outline-color*, *outline-style*, *outline-width* és *outline-offset* – ként, ill. az első három tulajdonság összevontan (a szegélyek mintájára, és ugyanazokkal az értékekkel) *outline*-ként adható meg.

Alapértelmezetten a szegély élén jelenik meg a körvonal (ha nincsen szegély, akkor is matematikailag létezik ez a hely 0 szegélyszélességgel), ezt a helyzetet lehet az *outline-offset*-el az elemhez közelebbre (negatív érték), vagy attól távolabbra (pozitív érték) eltolni.

Példaként körvonallal látunk el egy fekvő és egy álló ellipszist, ill. egy kört:

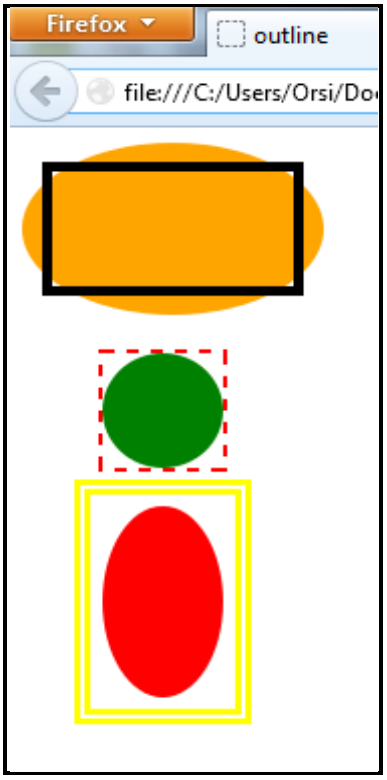
HTML:

```
<body>
     <div id="ovalisegy"></div><br>
     <div id="korong"></div><br>
     <div id="ovalisketto"></div>
</body>
```

CSS: (a *margin*-ok csak az egymás alá rendezést szolgálják)

```
<style>
   #ovalisegy { width:150px; height:90px; border-radius:50%;
                background-color:orange; outline:solid black 5px;
                outline-offset:-15px; }
   #korong { width:60px; height:60px; border-radius:50%;
                background-color:green; outline:dashed red 2px;
                outline-offset:0px; margin-left:40px; }
   #ovalisketto { width:60px; height:100px; border-radius:50%;
                background-color:red; outline:double yellow 8px;
                outline-offset:6px; margin-left:40px; }
</style>
```

A megjelenítés:



Mivel a körvonal nem befolyásolja olyan értelemben a formázást, hogy a dobozmodellben nincsen számára kihagyva hely, változatlanul a helyén maradó más szomszédos elemre is rákerülhet. Ha pl. kivesszük a HTML-kódból a <*br*>-eket, a sárga körvonal részben a zöld elemre esik, és kitakarja az alákerült részt.

Megjegyzés: Az Internet Explorer-ek nem értelmezik az *outline-offset* tulajdonságot, mindig a szegélyszélhez illesztik a körvonalat.

3.18. Blokkszintű elemek elhelyezése II.

3.18.1. Elhelyezés módja

A böngészők egy HTML dokumentum olvasásakor balról jobbra és fentről lefelé haladnak (ez a normál szövegfolyam), a HTML-elemek alapértelmezetten a kódolási sorrendnek és a normál szövegfolyamnak megfelelően vagy sorban, vagy blokkszerűen egymás alatt jelennek meg. Ebből a szerkezeti sémából egy adott elem az alábbi fejezetben ismertetett tulajdonságokkal ragadható ki és pozícionálható más szabályok szerint.

3.18.1.1. A **position** (elhelyezés) tulajdonsággal különböző viszonyítási pontokhoz képesti új elhelyezés állítható be. Az adott elem új helyzetét a *top* (felül), *left* (balra), *bottom* (alul) *és right* (jobbra) tulajdonságok értékeivel definiáljuk, melyek *px*-ben, *em*-ben vagy *%*-ban adhatók meg.

A *position* tulajdonság lehetséges értékei:

- *static* (statikus): ez az alapértelmezett érték; az elem a normál folyamban az eredeti helyén marad, ezért az új helyzetet definiáló tulajdonságok (*top, left, bottom, right*) nem értelmezettek

- *relative* (relatív vagy viszonyított): az elem továbbra is a normál szövegfolyamban helyezkedik el, de az alapértelmezett *static* helyzetből vízszintes és/vagy függőleges irányba eltoljuk a *left, right, top, bottom* tulajdonságok valamilyen értékeivel. Az eltolt elem „eredeti helye" üresen marad, az eltolás a környezetére nincsen hatással (eltekintve attól, ha az új helyén egy másik elemmel esetleg takarják egymást – ekkor a kódolási sorrend dönti el, melyikük lesz „alul", azaz melyikük kerül takarásba).

- *absolute* (abszolút vagy független): az adott elem helyzete a tárolótömbjéhez képest van a *top, left, bottom, right* tulajdonságok értékeivel pozícionálva. A tároló lehet egy másik elem, vagy a kezdeti tárolótömb (maga a *html vagy* a *body* elem). A szövegfolyamban bárhova elhelyezhető a HTML-kódolás, úgyis adott lesz az elemnek a definiált koordináták szerinti helye (esetleg takarási viszonyba kerül majd az ott lévő tartalommal).

- *fixed* (rögzített): a weblap görgetése során is a felhasználó által látható böngészőablak részhez rögzített marad az elem. Az abszolút pozíció speciális esetének tekinthető, csak nem a befoglaló elem hanem a képernyő a fix viszonyítási pont.

A *position* tulajdonság különböző értékei egy egyszerű, három négyzetet és egy rövid szöveget tartalmazó weblapon jól bemutathatók. A kiinduló – pozícionálás nélküli – kódolás:

HTML:

```
<body>
      <div id="egy"></div>
      <div id="ketto"></div>
      <div id="harom"></div>
      <p>Lorem ipsum dolor sit amet……</p>
</body>
```

CSS:

```
<style>
       #egy { background-color:yellow; width:150px; height:150px; }
       #ketto { background-color:green; width:100px; height:100px; }
       #harom { background-color:blue; width:50px; height:50px; }
</style>
```

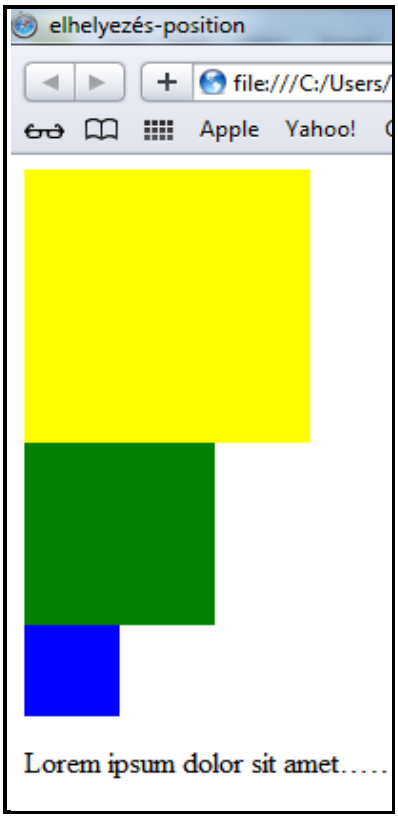
a) Ha a négyzetek tulajdonságait bővítjük *position*:*static;*-al, és egyenként *top:50px; top:-50px; top:150px;* helyzeteket definiálunk hozzájuk, akkor a kódolás:

```
<style>
     #egy { background-color:yellow; width:150px; height:150px;
            position:static; top:50px; }
    #ketto { background-color:green; width:100px; height:100px;
            position:static; top:-50px; }
    #harom { background-color:blue; width:50px; height:50px;
            position:static; top:150px; }
</style>
```

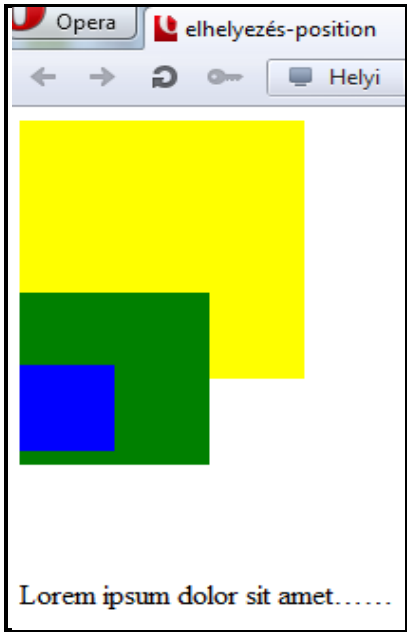
A megjelenítésben nem történik semmilyen változás, hiszen amúgy is a *static* az alapértelmezett érték, és ebben az esetben a *top, left, bottom* és *right* tulajdonságok nem értelmezettek.

b) Változtassuk meg a második és harmadik négyzet elhelyezési módját *relative*-re, ill. *absolute*-ra:

```
#egy { background-color:yellow; width:150px; height:150px;
       position:static; top:50px; }
#ketto { background-color:green; width:100px; height:100px;
       position:relative; top:-50px; }
#harom { background-color:blue; width:50px; height:50px;
       position:absolute; top:150px; }
```



a) alapértelmezett



b) *relative* és *absolute*

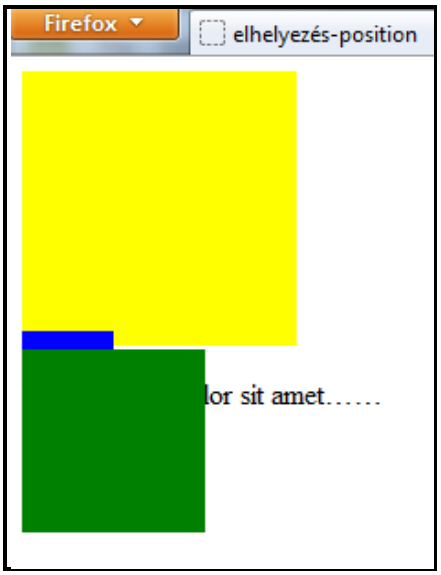
A *static* négyzet helyzete nem változott. A *relative* négyzet az eredeti helyzetéhez képest *50px*-el függőleges irányban eltolódott (ha pozitív *50px* lett volna megadva, lefelé tolódott volna - negatív értékről lévén szó, felfelé mozdult el). Az „eredeti helye” üresen maradt, a szöveg nem került oda. Az *absolute* négyzet elhelyezése most a *body* bal felső sarkától számítódik, így a látszólagos függőleges lefelé tolás ellenére feljebb került, mert a viszonyítási pont megváltozott (korábban a felette lévő *div*-hez igazodott). A HTML-kódolás szerinti helye a többi elem szempontjából nem létezik. Az új pozíciók következtében keletkezett takarások a kódolás sorrendjét követik – a későbbi kódolású tartalom takarja a korábbit.

c) Cseréljük meg a HTML-kódok sorrendjét változatlanul hagyva a CSS-kódolást:

```
<div id="egy"></div>
<p>Lorem ipsum dolor sit amet……</p>
<div id="harom"></div>
<div id="ketto"></div>
```

A *static* négyzetnél nem lesz változás, mert sem a kódolásban elfoglalt helye, sem a helyzete nem változott. A szöveg követi a sárga négyzetet a kódolás szerint. A zöld négyzet relatív helyzete a szöveghez képest értendő, így lejjebb kerül, már nem ér el a sárga négyzetig, viszont takarja a szöveget, miután utána következik a normál szövegfolyamban. A kék abszolút pozícióját nem érinti a sorrendváltozás, viszont most takarja a zöld, mert a kéknél később következik a HTML-kódolásban.

A megjelenítés tehát átalakult az alábbiak szerint:



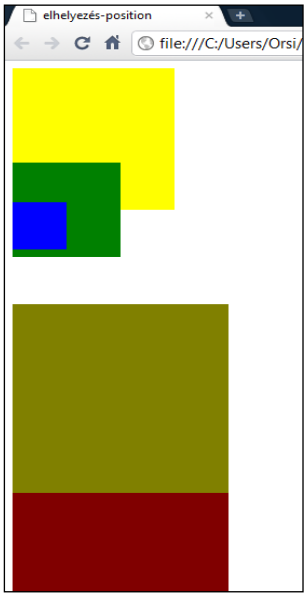
d) A *fixed* elhelyezéshez olyan nagyra kell a tartalmat növelni, hogy görgetősáv jelenjen meg, és a tartalom görgethetővé váljon. Ezért hozzáteszünk még négy nagy négyzetet az eredeti háromhoz, és a kék helyzetét *fix*-re módosítjuk (a szöveg most nem érdekes):

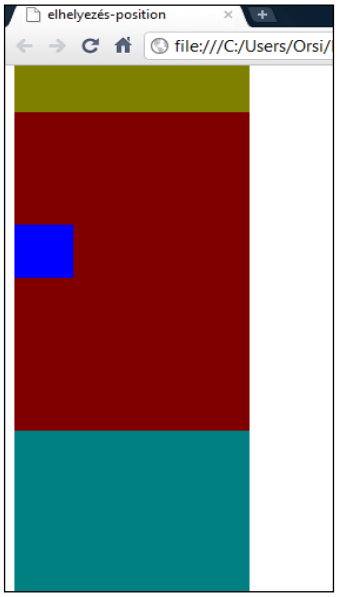
```
<style>
   #egy { background-color:yellow; width:150px; height:150px;
          position:static; top:50px; }
   #ketto { background-color:green; width:100px; height:100px;
          position:relative; top:-50px; }
   #harom { background-color:blue; width:50px; height:50px;
          position:fixed; top:150px; }
   #negy { background-color:olive; width:200px; height:200px;
          position:static; top:50px; }
   #ot { background-color:maroon; width:200px; height:300px;
          position:static; top:50px; }
   #hat { background-color:teal; width:200px; height:300px;
         position:static; top:50px; }
   #het { background-color:black; width:200px; height:300px;
         position:static; top:50px; }
</style>
<body>
    <div id="egy"></div>
    <div id="ketto"></div>
    <div id="harom"></div>
    <div id="negy"></div>
    <div id="ot"></div>
    <div id="hat"></div>
    <div id="het"></div>
</body>
```

Miután alapértelmezett *static* elhelyezésűek az új négyzetek, egymás alatti blokkokként jelennek meg. A *relative* (zöld) négyzet „eredeti helyét” nem foglalják el, az *absolute* (kék) négyzet helyzete a koordinátáival adott, és minden érintett négyzetet takar.

Az új négyzetek túllógnak a böngészőablak függőleges méretén, tehát van görgetősáv. A görgetősávot mozgatva a kék kocka a rögzített helyén marad, a többi kocka a görgetősávval együtt mozog.

Egy pillanatfelvétel a görgetés közbeni állapotról:





Megjegyzés: Csak az *absolute* elhelyezési módú elemekre alkalmazható a *clip* (levágás) tulajdonság, mellyel az definiálható, hogy az adott elem szegélydobozának mekkora része legyen látható.

Lehetséges értékei:

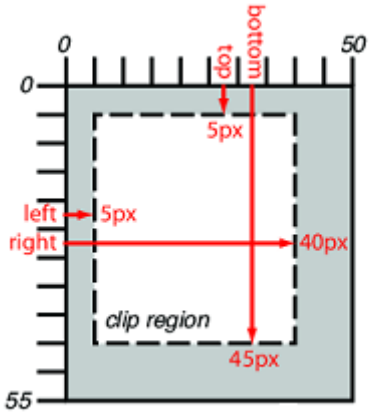
- *auto*: nincsen megadva semmilyen levágás, a szegélydoboz teljes mértékben látható

- *shape* (alak): a látható rész adható meg - csak négyzet alak definiálható, melynek négy sarkát az elem szegélydoboza bal felső sarkától számítva, *px*-ben, felsorolásszerűen, az óramutató járásával megegyező irányban, vesszővel és szóközzel elválasztva lehet megadni.

A *clip* tulajdonság az adott elem valamennyi komponensére hat, azaz a kijelölt részen (*clipping region*) kívüli tartalom, háttér, görgetősáv, körvonal, stb. nem fog látszani.

Például ha egy *50* x *55 px*-es elemnek a felső részéből és bal széléből *5-5px*-nyi, a jobb oldalából és az alsó széléből *10-10px*-nyi részt le akarunk vágni, akkor a szaggatottal jelölt terület marad látható, és a kódolás:

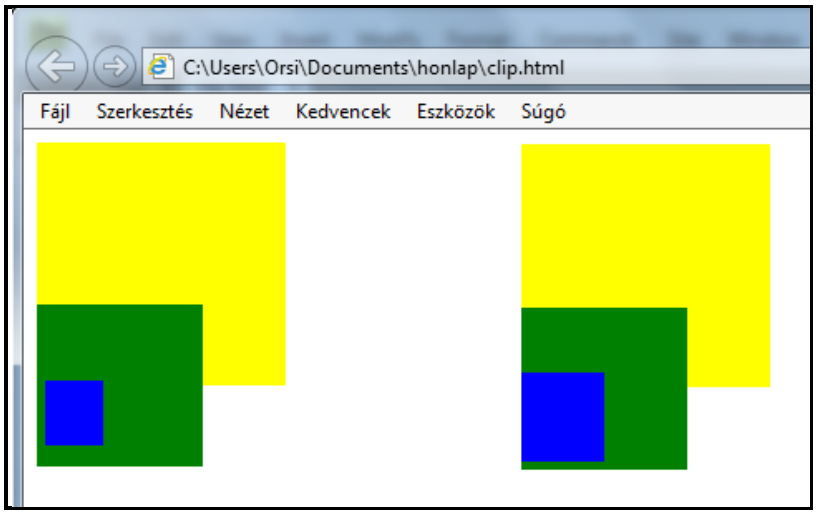
```
#adottelem { clip: rect(5px, 40px, 45px, 5px); }
```



A három négyzetes korábbi CSS-kódunkat módosítva a kivágással:

```
#egy { background-color:yellow; width:150px; height:150px;
       position:static; top:50px; }
#ketto { background-color:green; width:100px; height:100px;
       position:relative; top:-50px; }
#harom { background-color:blue; width:50px; height:55px;
       position:absolute; top:150px; clip: rect(5px, 40px,
       45px, 5px); }
```

A kivágott megjelenítés és összehasonlításként jobb oldalt a kivágás nélküli állapot:



3.18.1.2. A **float** (úsztatás) tulajdonsággal vízszintes irányban a befoglaló elem széleihez úsztathatók (tolhatók) elemek. Az *inline* elemeknek is blokk-dobozt hoz létre, a szomszédos elemekhez képesti távolság a *margin* tulajdonsággal definiálható.

A *float* lehetséges értékei (függőlegesen nem lehet úsztatni):

none : ez az alapértelmezett – nincsen úsztatás
left : a képet a weblap bal szélére úsztatja, azaz tolja
right : a képet a weblap jobb szélére úsztatja, azaz tolja

Az úsztatott elemet követő elemek is úszni fognak, ezt a **clear** (mezőkiürítés, törlés, úsztatott elem mögüli eltávolítás) tulajdonsággal lehet megszüntetni, melynek lehetséges értékei:

none: nincsen törlés – ez az alapértelmezett
left: törlés balra
right: törlés jobbra
both: törlés mindkét irányba

A *position* tulajdonságnál használt három négyzettel bemutatható a *float* és *clear* hatása is. Az úsztatás előtti állapot kódolása megegyezik az ottani kiinduló állapottal:

HTML:

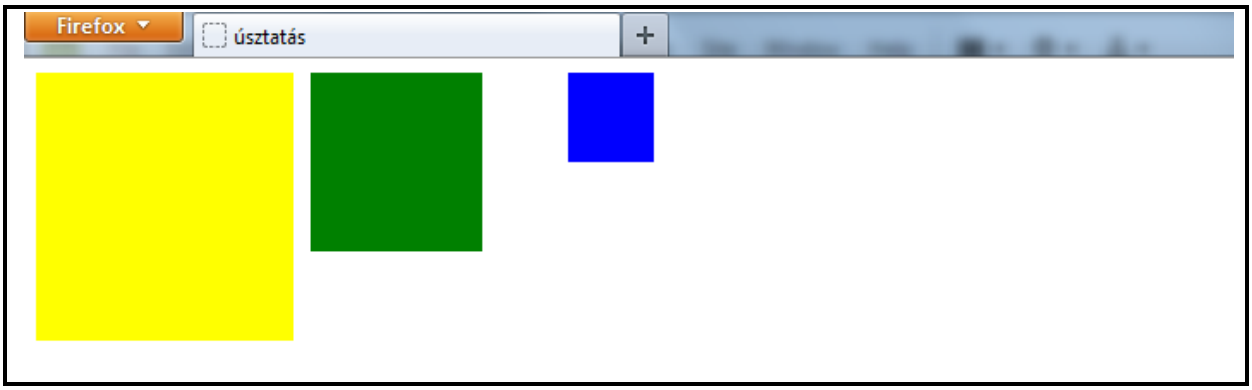
```
<body>
      <div id="egy"></div>
      <div id="ketto"></div>
      <div id="harom"></div>
      <p>Lorem ipsum dolor sit amet……</p>
</body>
```

CSS:

```
<style>
  #egy { background-color:yellow; width:150px; height:150px; }
 #ketto { background-color:green; width:100px; height:100px; }
  #harom { background-color:blue; width:50px; height:50px; }
</style>
```

a) Legyenek először a négyzetek balra úsztatva – ekkor mind egy sorba, egymástól a megadott margónak megfelelő távolságra kerülnek:

```
#egy { background-color:yellow; width:150px; height:150px;
          float:left; }
#ketto { background-color:green; width:100px; height:100px;
          float:left; margin-left: 10px; }
#harom { background-color:blue; width:50px; height:50px;
          float:left; margin-left: 50px; }
```

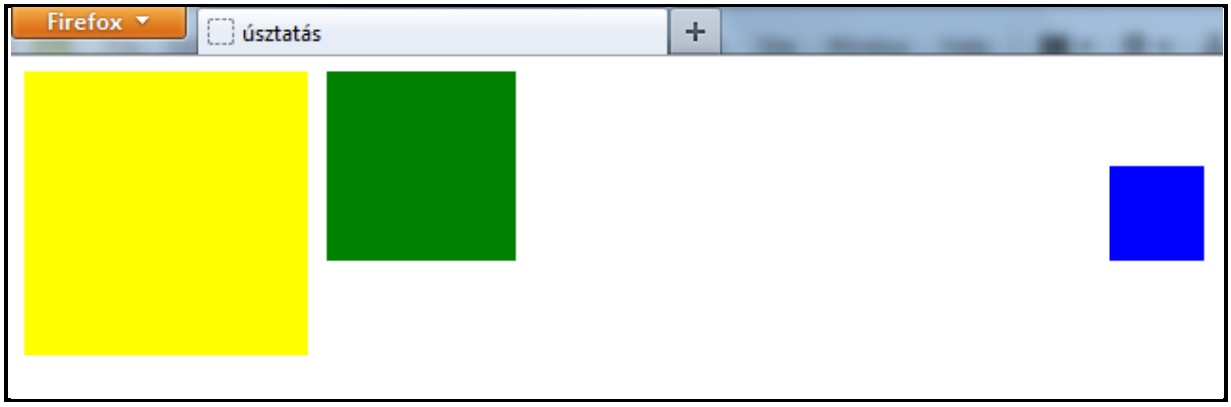


A sárga négyzet helyzete nem változott, mivel az addig is balra volt igazítva, viszont az őt követő elemek a jobb oldalán úsznak a *margin*-ok szerinti elhelyezkedéssel.

b) Kódoljunk a kék négyzetnek jobbra úsztatást és valamilyen felső margót (a két másik négyzet kódolása maradt változatlan):

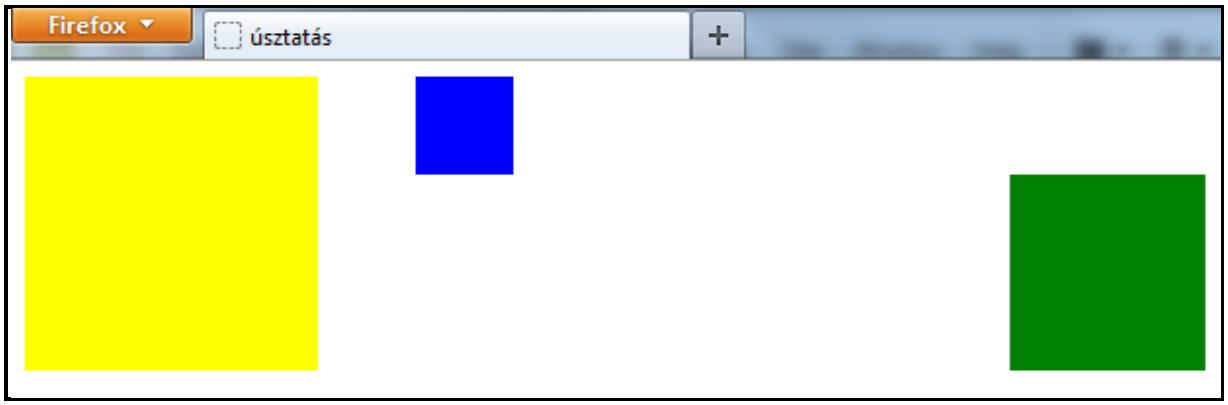
```
#harom { background-color:blue; width:50px; height:50px;
          float:right; margin-top: 50px;}
```

A várakozásnak megfelelően a kék négyzet a jobb szélre tolódik és a megadott felső margónak megfelelően lejjebb kerül:



c) Kódoljunk a zöld négyzetnek jobbra úsztatást és valamilyen felső margót (a két másik négyzet eredeti úsztatott – a) - kódolása maradt változatlan):

```
#ketto { background-color:green; width:100px; height:100px;
float:right; margin-top:50px; }
```

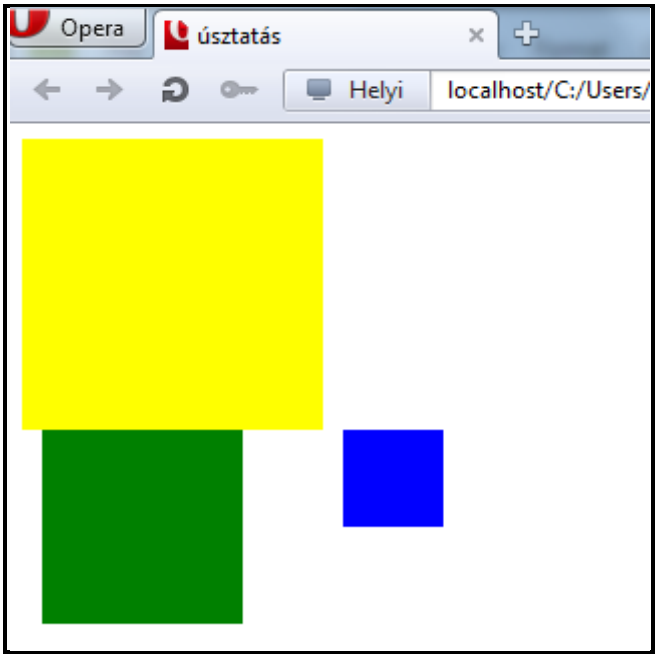


Az úsztatás következtében a zöld négyzet a befoglaló elem (*body*) jobb széléhez igazodott, ugyanakkor kikerülve a normál szövegfolyamból, a HTML-kódolás sorrendje ellenére a kék kocka mögött jelent meg.

d) A zöld négyzetnek adjunk meg egy *clear:left;* tulajdonságot/értéket (a többi marad az a) pont szerint):

```
#ketto { background-color:green; width:100px; height:100px;
         float:left; margin-left: 10px; clear:left; }
```

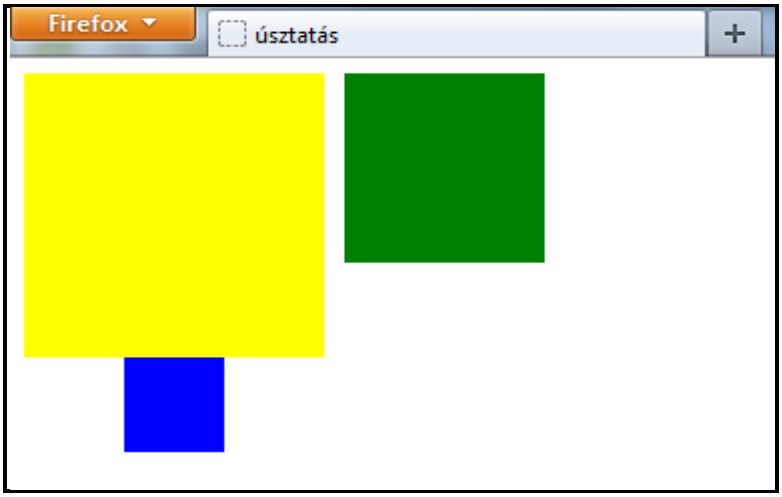
Az úsztatás a zöld négyzet bal oldalán a sárga négyzethez képest megszűnik, ezért a zöld négyzet a normál szövegfolyam szerinti blokkszintű elemként új sorban helyezkedik el, és annak bal széléhez igazodik. Jobb oldalán ugyanakkor továbbra is érvényes maradt az úsztatás, a kék kocka az új sorban igazodik a zöldhöz a *float* szerint:



e) A kék négyzetnek adjunk meg egy *clear:left;* tulajdonságot/értéket (a többi marad az a) pont szerint):

```
#harom { background-color:blue; width:50px; height:50px;
         float:left; margin-left: 50px; clear:left; }
```

Az úsztatás a kék négyzet bal oldalán a zöld négyzethez képest megszűnik, ezért a kék négyzet a normál szövegfolyam szerinti blokkszintű elemként új sorban helyezkedik el, és annak bal széléhez – a margót is figyelembe véve - igazodik. A zöld négyzet továbbra is úszik a sárga négyzet bal oldalánál.



Elvileg minden elemet lehet úsztatni, a gyakorlatban a képeknek/táblázatoknak szöveggel való körbefuttatásánál, iniciálék kialakításánál vagy a weboldal fő részeinek elhelyezésére szokták a *float*-ot és *clear*-t a leggyakrabban alkalmazni.

Ha pl. egy kép úsztatva van, az őt tartalmazó befoglaló szövegblokk adott sorának a jobb vagy bal széléig tolódik, a tartalom többi része (a szöveg) pedig körbefolyja. A *float* és a körbefutó tartalom közötti helykihagyás a *float*-nak a *margin* tulajdonságával változtatható.

Példaként a *lipsum*-ba egy tetszőleges helyre a közepe felé berakunk egy üres elemet (ami helyettesíthet képet vagy más elemet) - a HTML-kódolás:

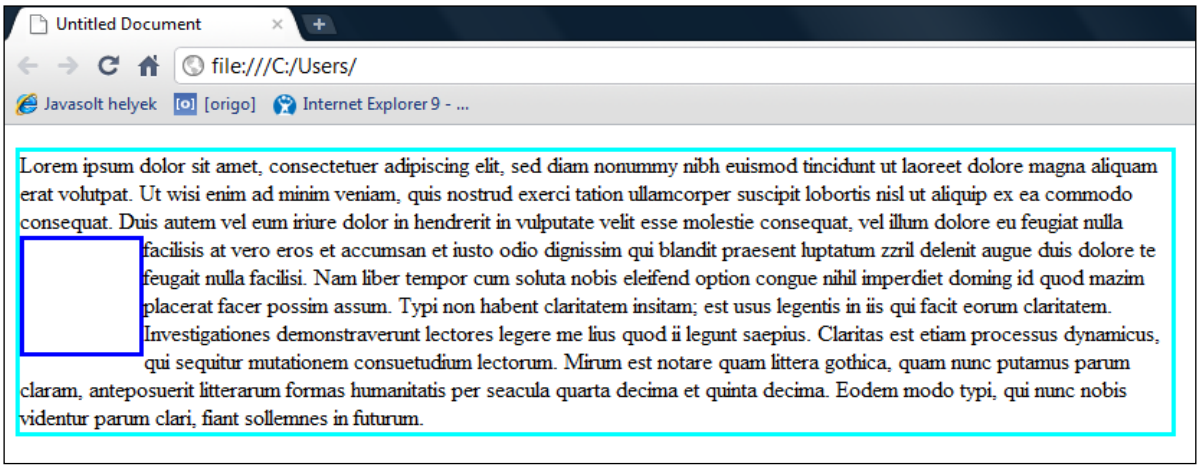
```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore
magna aliquam erat volutpat. Ut wisi enim ad minim veniam,
quis nostrud exerci tation ullamcorper suscipit lobortis nsl
út aliquip ex ea commodo consequat. Duis autem vel eum iriure
dolor in hendrerit in vulputate velit esse molestie consequat,
vel illum dolore eu feugiat nulla <span></span> facilisis at
vero eros et accumsan et iusto odio dignissim qui blandit
praesent luptatum zzril delenit augue duis dolore te feugait
nulla facilisi. Nam liber tempor cum soluta nobis eleifend
option congue nihil imperdiet doming id quod mazim placerat
facer possim assum. Typi non habent claritatem insitam; est
usus legentis in iis qui facit eorum claritatem. Investiga-
tiones demonstraverunt lectores legere me lius quod ii legunt
saepius. Claritas est etiam processus dynamicus, qui sequitur
mutationem consuetudium lectorum. Mirum est notare quam lit-
tera gothica, quam nunc putamus parum claram, anteposuerit
litterarum formas humanitatis per seacula quarta decima et
```

```
quinta decima. Eodem modo typi, qui nunc nobis videntur parum
clari, fiant sollemnes in futurum.</p>
```

A CSS-kódban az úsztatott elem legyen bal oldalra tolva, alapértelmezett 0 margóval:

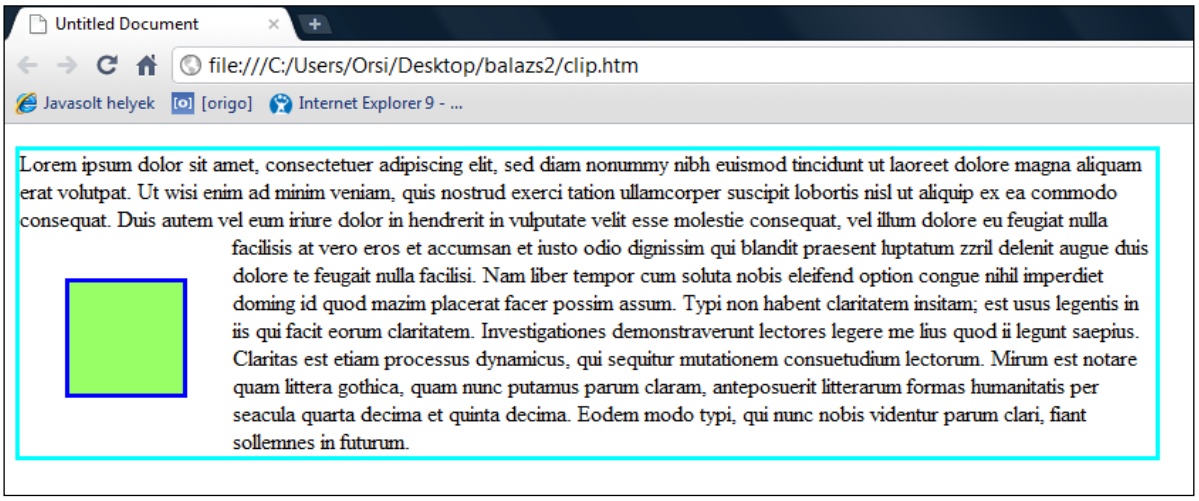
```
p { width:50em; border:solid aqua; }
span { float:left; width:5em; height:5em; border:solid blue; }
```

A megjelenítés az alábbi lesz:



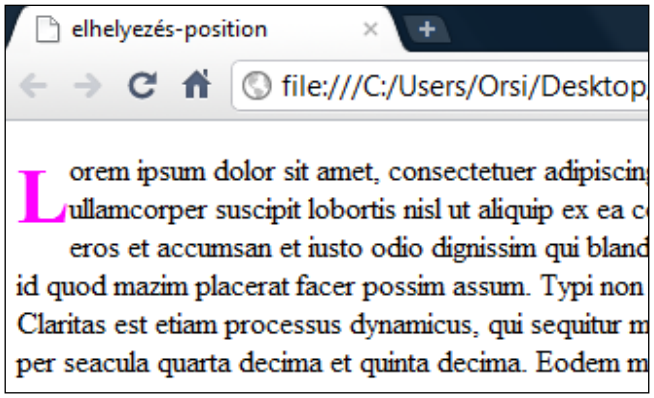
A CSS kódolásban margóval távolabbra állítva a körbefolyó szöveget és háttérszínt adva a *float* elemnek:

```
p { width: 50em; border: solid aqua; }
span { float:left; width:5em; height:5em; border:solid blue;
       margin:2em; background-color:#99FF66; }
```



A *::first-letter* és *float* együttes használatával **iniciálé**k (*drop cap initial letter*) alakíthatók ki. Maradva a fenti *lipsum* bekezdésnél, elhagyva a *float* demonstrálására bekódolt négyzetet belőle, és az első betűnket iniciálának formázva:

```
::first-letter { font-size:250%; font-weight:bold;
                 color:fuchsia; font-family:Corsiva; float: left; }
```



3.18.2. Láthatóság

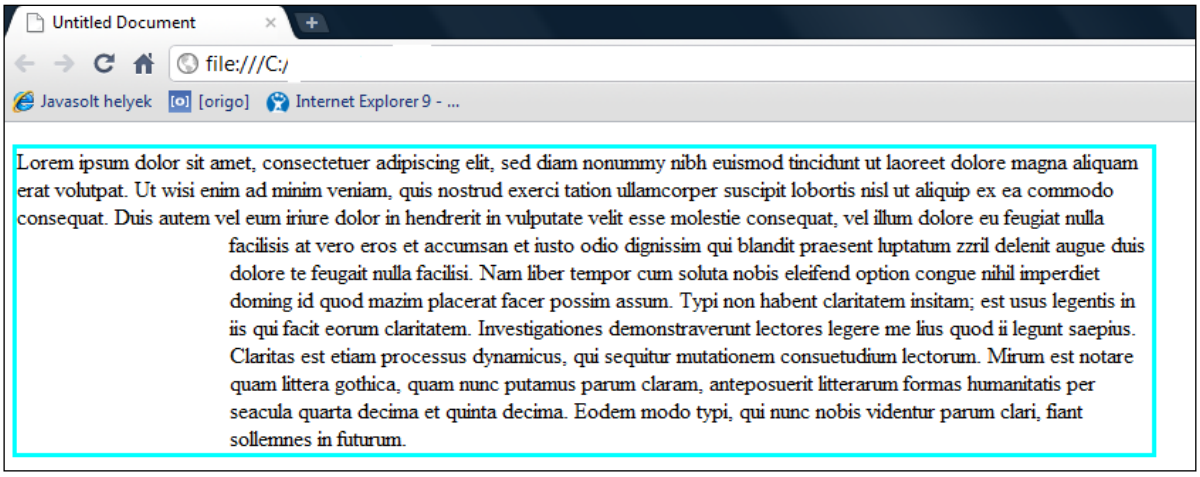
A **visibility** (láthatóság) tulajdonság egy adott elem két lehetséges állapotát jelöli, nevezetesen, hogy látható vagy láthatatlan legyen. Alapértelmezett értéke *visible* (látható), a másik érték *hidden* (rejtett). A láthatatlan (rejtett) elem a weblap elrendezésében megtartja a helyét, csak átlátszóvá válik.

Általában ugyanazon a helyen egymásra helyezett elemek láthatóságának a látogató valamilyen akciója révén kiváltott változtatására használják (lásd pl. az *Egérkurzorral a megjelenítés megváltoztatása* fejezetet), de a weblap kialakítása során egyébként is hasznos lehet egyes elemek időleges eltakarása.

Az előző példa úsztatott dobozát pl. el tudjuk rejteni az alábbi CSS kód hozzáadásával:

```
span { float:left; width:5em; height:5em; border:solid blue;
   margin:2em; background-color: #99ff66; visibility:hidden; }
```

(Az elrejtett elem továbbra is ott marad, és a *hidden* érték megváltoztatásával bármikor ismét láthatóvá tehető.)

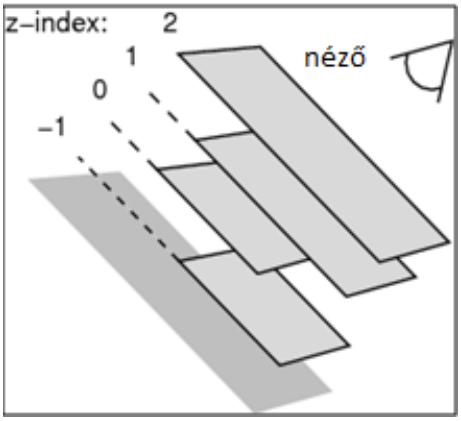


3.18.3. Láthatósági sorrend

Az egymást takaró elemek láthatóságát alapértelmezetten a HTML-dokumentumban elfoglalt sorrendjük határozza meg (a későbbi elem takarja a korábbit). A láthatósági sorrendet a **z-index** tulajdonsággal lehet megváltoztatni, mely az elem síkjára merőleges *z*-

*tengely*ről kapta a nevét. (Az *x-tengely* a vízszintes pozíciót, az *y-tengely* a függőleges pozíciót definiálja egy elem síkjában, az ezekre merőleges (virtuális) *z-tengely* harmadik dimenziót, térbeliséget szimbolizál. A gyakorlatban a használata a *visibility* tulajdonságnál említett, a látogató valamilyen akciója révén kiváltott (pl. *:hover* vagy *:active*) hatással történik.

A *z-index*-nél (tetszőleges előjelű egész) számok növekvő sorrendjébe rendeződik a láthatóság – a legkisebb index-számú elem kerül legalulra (a nézőtől a legtávolabbra), a legnagyobb index-számú a legfelülre (a nézőhöz a legközelebb). Nem kell egymást követő számoknak lenniük, és bárhonnan indulhat a számozás.



Egy egyszerű példát egy ugyanarra a helyre helyezett zöldes háttérszínű és lilás háttérszínű, ugyanakkora méretű négyszögön is be lehet mutatni:

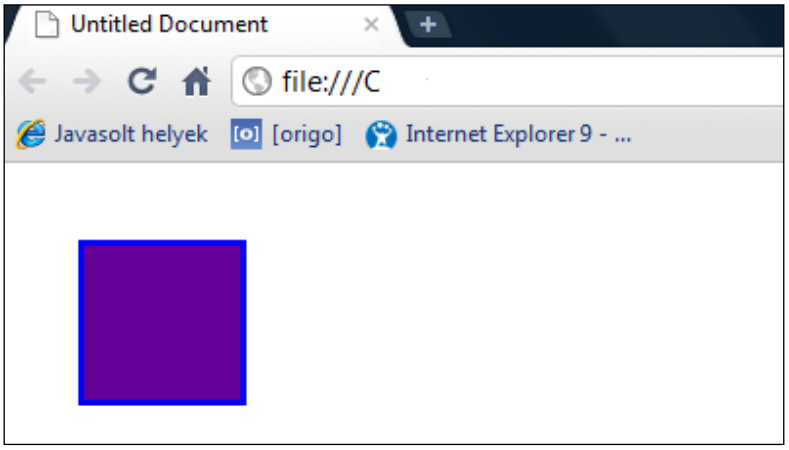
HTML-kódjuk:

```
<div id="egy"></div>
<div id="ketto"></div>
```

CSS-kódjuk:

```
#egy { position:absolute; width:5em; height:5em; border:solid
     blue; margin:2em; background-color:#99FF66; }
#ketto { position:absolute; width:5em; height:5em;
    border:solid blue; margin:2em; background-color:#660099; }
```

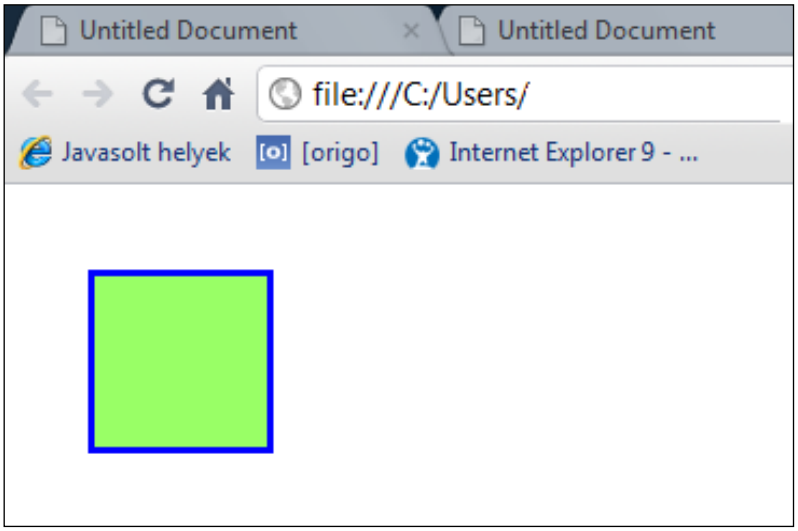
A megjelenítése:



Az egymásra helyezett négyzetek közül alapértelmezetten a HTML-kódban későbbi takarja az előbbit, tehát a lilás hátterű látszik.

A CSS-be *z-index*-eket beírva (a zöldes hátterű indexe legyen a nagyobb):

```
#egy { position:absolute; width:5em; height:5em; border:solid
       blue; margin:2em; background-color:#99ff66; z-index:1; }
#ketto { position:absolute; width:5em; height:5em;
       border:solid blue; margin:2em; background-color:#660099;
       z-index:0; }
```



A *z-index* hatására a zöldes hátterű négyszög került felülre és takarja a lilás hátterűt.

3.18.4. A megjelenítés módja

A **display** (megjelenítés) tulajdonság szabja meg az elemhez generált doboz típus(o-ka)t. Előírhatjuk vele, hogy egy elem milyen módon jelenjen meg, lehetséges értékei: *table, inline-table, table-row-group, table-header-group, table-footer-group, table-row, table-cell, table-column-group, table-column, table-caption, run-in, list-item, flexbox, inline-flexbox, none, inline, block, inline-block.*

A leggyakrabban használt értékek (melyeket mindegyik böngésző értelmez):

- *inline* (szövegközi vagy folytonos elrendezésű) – ez az alapértelmezett érték
- *block* (blokk doboz)
- *none* – az elemhez nem rendel dobozt, azaz az elemnek nincsen hatása a weboldal elrendezésére (nem egyszerűen láthatatlan mint a *visibility:hidden* használatakor, amely viszont befolyásolja a weboldal elrendezését)

a) Alapértelmezetten blokk-elemből folytonos elrendezésű elem alakítható ki a *display* tulajdonság *inline* értékével. Pl. egy lista vízszintes kialakítása (ami leggyakrabban egy vízszintes menü-sor kialakításánál használatos):

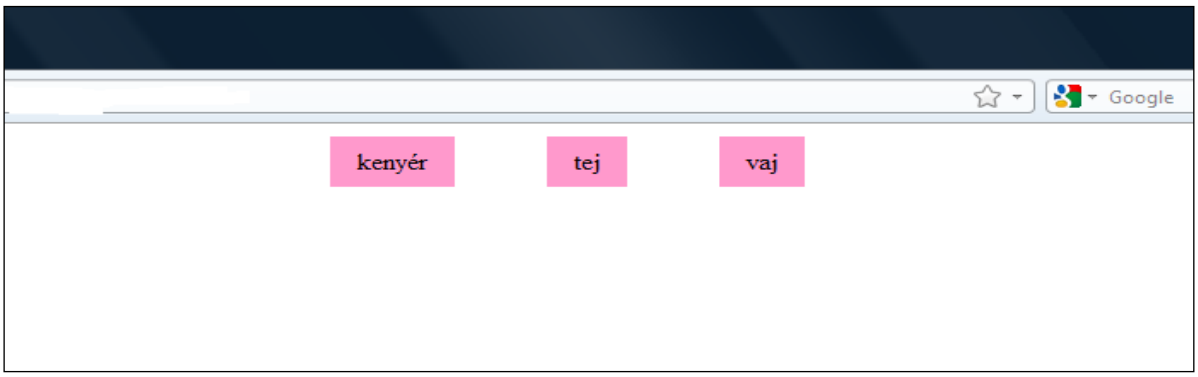
HTML-kód:

```
<ul>
     <li>kenyér</li>
     <li>tej</li>
     <li>vaj</li>
</ul>
```

A CSS kódolása:

```
li { display:inline; list-style:none; margin-right:50px;
     background-color:#FF99CC; padding:0.5em 1em; }
ul { text-align:center; }
```

A *list-style:none*-al tüntettük el a felsorolásjeleket, a *margin-right:50px*-el helyeztük távolabb egymástól a lista-elemeket, *háttérszín*t a jobb láthatóság érdekében definiáltunk, a *padding*–el a háttér szélét vittük a szövegtől kissé messzebb, a *text-align*–al pedig a fentiek szerint formázott listát helyeztük középre:



b) Alapértelmezetten sorközi elemből blokk-elem alakítható ki a *display* tulajdonság *block* értékével. Pl. egy kép (mely alapértelmezetten *inline* elem) blokkszintű elemmé alakítható, amely természetesen megváltoztatja a tartalomban való megjelenítését.

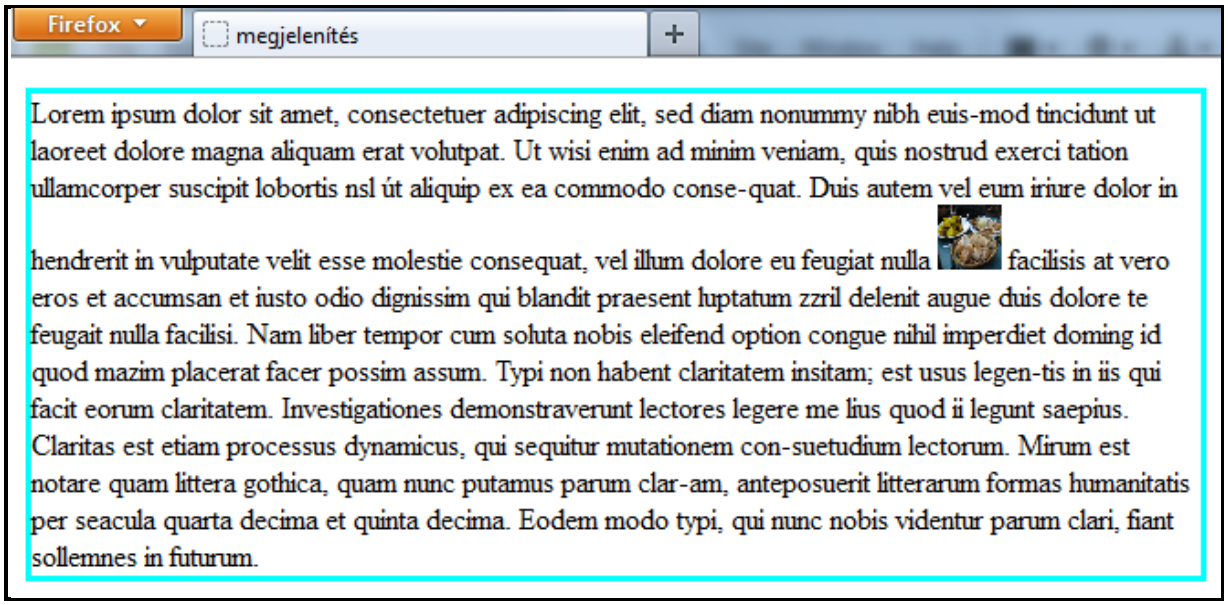
Egy szövegben elhelyezett alapértelmezett kép HTML-kódja:

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore
magna aliquam erat volutpat. Ut wisi enim ad minim veniam,
quis nostrud exerci tation ullamcorper suscipit lobortis nsl
út aliquip ex ea commodo consequat. Duis autem vel eum iriure
dolor in hendrerit in vulputate velit esse molestie consequat,
vel illum dolore eu feugiat nulla <img src="IMG_1937.JPG"
width="35px" height="35px"> facilisis at vero eros et accumsan
et iusto odio dignissim qui blandit praesent luptatum zzril
delenit augue duis dolore te feugait nulla facilisi. Nam liber
tempor cum soluta nobis eleifend option congue nihil imperdiet
doming id quod mazim placerat facer possim assum. Typi non
habent claritatem insitam; est usus legentis in iis qui facit
eorum claritatem. Investigationes demonstraverunt lectores
legere me lius quod ii legunt saepius. Claritas est etiam
processus dynamicus, qui sequitur mutationem consuetudium
lectorum. Mirum est notare quam littera gothica, quam nunc
putamus parum claram, anteposuerit litterarum formas
humanitatis per seacula quarta decima et quinta decima. Eodem
modo typi, qui nunc nobis videntur parum clari, fiant
sollemnes in futurum.</p>
```

A bekezdés CSS-kódja:

```
p { width:40em; border:solid aqua; }
```

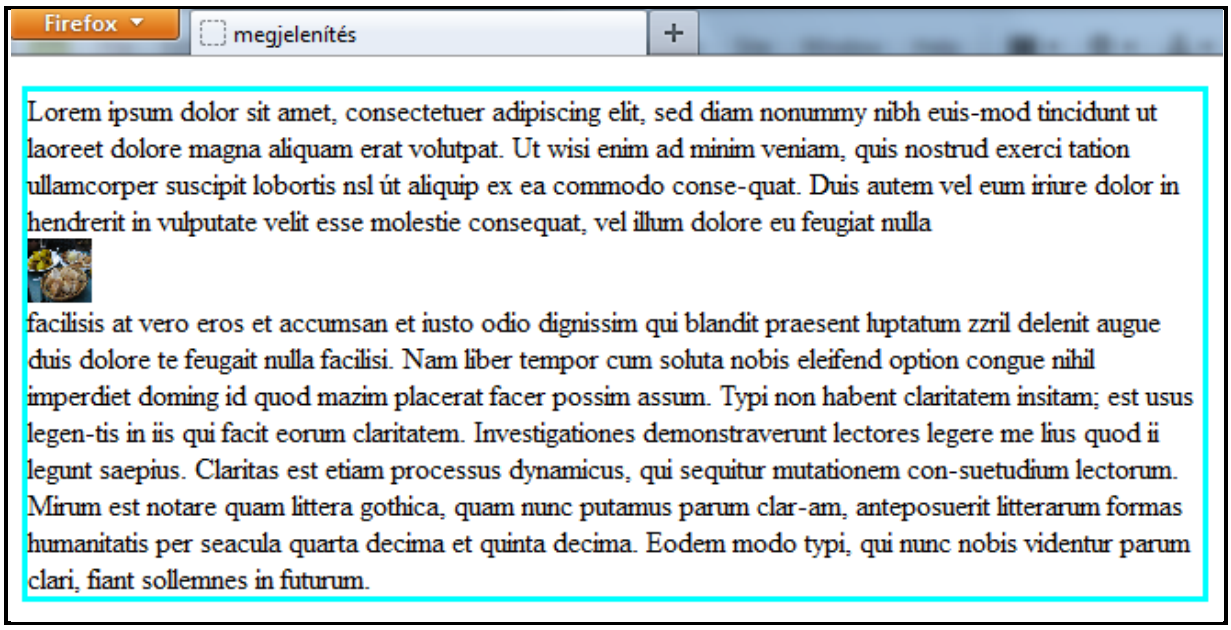
A megjelenítés (a képet tartalmazó sor függőleges igazításával nem foglalkoztunk):



A képet blokk-elemmé alakítva a CSS-kódolás:

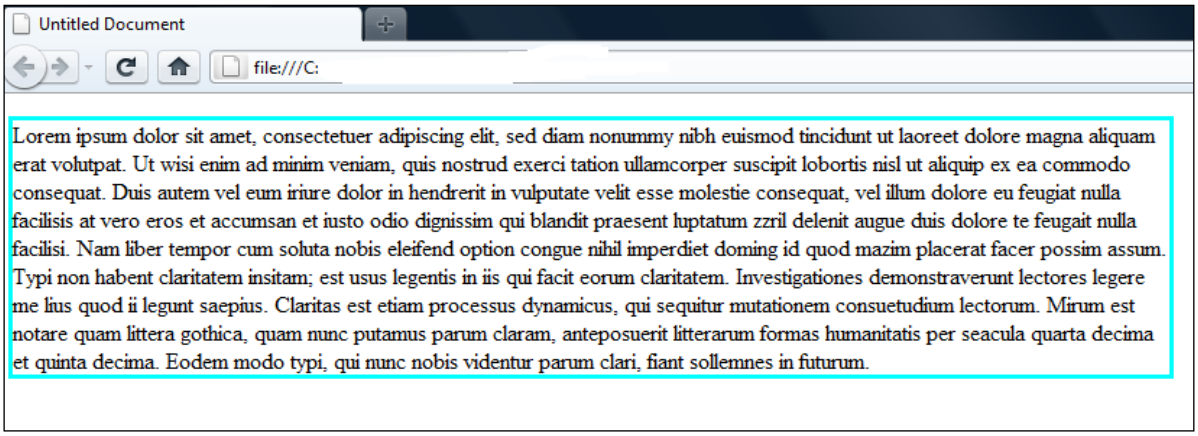
```
p { width:40em; border:solid aqua; }
img { display:block; }
```

Blokk-elemmé válva a kép külön sorban foglal helyet, és a továbbiakban másképpen formázandó (pl. más megjelenítést eredményez ugyanannak a *margin*-nak a bekódolása:



c) A korábbi úsztatott elemes példára (3.18.2. pont) alkalmazva a *display*:*none* definíciót a CSS kód az alábbi lesz:

```
p { width:50em; border:solid aqua; }
span { float:left; width:5em; height:5em; border:solid blue;
       margin:2em; background-color:#99ff66; display:none; }
```



A *none* érték hatására a négyzet elemnek a helye is eltűnik, a weboldal megjelenítés szempontjából figyelmen kívül hagyja az így formázott elemet.

Megjegyzés: A fejezetben ismertetett tulajdonságokkal nagyfokú szabadság érhető el az elemek egymástól függetleníthető elhelyezésében, mozgatásában, fedési sorrendjüknek és láthatóságuknak a kialakításában. Ezek a lehetőségek hasonlóak a képszerkesztő programoknak a független rétegekkel végezhető műveleteihez.

A blokkszintű elemek elhelyezésére és láthatóságára felhasznált CSS tulajdonságok:

position	elhelyezés módja
top	felül
left	balra
bottom	alul
right	jobbra
float	úsztatás
clear	törlés, mezőkiürítés, úsztatott elem mögüli eltávolítás
visibility	láthatóság
z-index	fedési sorrend
display	megjelenítés

3.19. Doboz árnyékok

A **doboz-árnyék** (*box-shadow*) tulajdonság egy vagy több vetett árnyékot csatol egy dobozhoz. A külső árnyék a szegélyen kívülre vetődik, mintha a szegélydoboz átlátszatlan lenne. A belső árnyék a belső margó szélétől befelé vetődik, mintha attól kifelé lenne minden átlátszatlan. Ha a doboznak lekerekített szegélyei vannak, az árnyék alakja is követi a lekerekítéseket.

A doboz-árnyék tulajdonság árnyékok vesszővel elválasztott listájából áll, melyben mindegyik árnyék 2-4 db hosszúság értéket, színt és opcionális *inset* kulcsszót tartalmaz. A kihagyott hosszúságok 0-nak értendők, ha nincsen szín megadva, a böngésző a szabvány szerint a *color* aktuális értékét rendeli hozzá (a gyakorlatban adjuk meg mindíg a színt !).

Az egyes árnyék-komponensek az alábbiak:

- az első hosszúság az árnyék vízszintes kiterjedése: pozitív értéknél a doboztól jobbra, negatív értéknél a doboztól balra rajzolódik árnyék
- a második hosszúság az árnyék függőleges kiterjedése: pozitív értéknél lefelé, negatív értéknél felfelé rajzolódik az árnyék
- a harmadik hosszúság az életlenítési (*blur*) távolság: negatív érték nem megengedett, ha 0 az értéke, akkor az árnyék széle éles, egyébként pedig minél nagyobb pozitív érték, annál inkább eléletlenedik az árnyék széle
- a negyedik hosszúság a szórás/kiterjedés/nyílásszög (*spread*) távolság: pozitív értéknél az árnyék alakja minden irányban kiterjed a megadott sugárban, negatív értéknél összehúzódik
- az árnyék színe (*color*-nál felsorolt megadási módokkal)
- az *inset* kulcsszó változtatja meg a külső árnyékot belső árnyékká

Több árnyék esetén elölről hátrafelé haladva látszódnak: az első van legfelül, a többiek alárétegződnek. Az árnyék átnyúlhat más dobozokba, vagy más dobozok árnyékaiba. Egy elem belső árnyékai az elem háttere fölé és a szegélye alá, a külső árnyékok az elem háttere alá kerülnek.

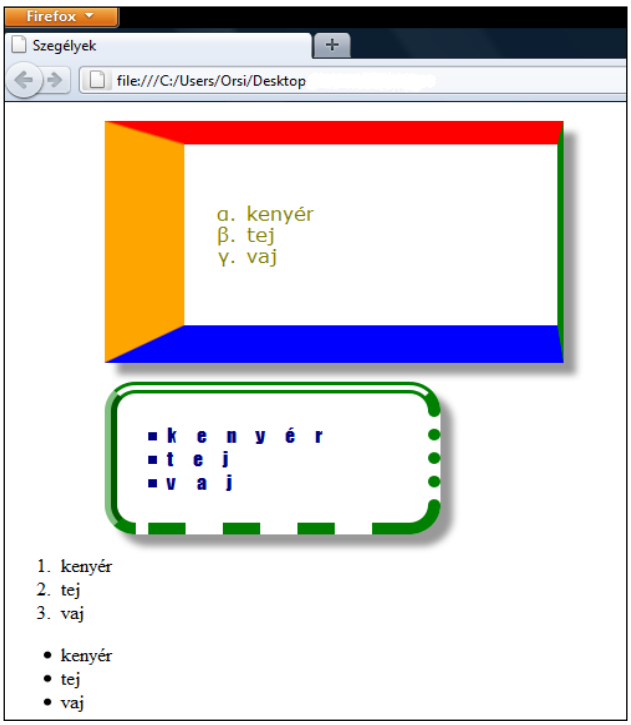
Megjegyzés: A doboz-árnyékok kódolása a szöveg-árnyékkal analóg módon történik, csak két paraméterrel (*spread, inset*) szélesebb a választék.

Figyelem! Az Internet Explorer 8 árnyék nélkül jeleníti meg az adott elemet.

Formázott és bekeretezett listáinkat szürke, ferde, életlenített külső árnyékkal látjuk el. Az árnyék szürke színét a fekete részleges átlátszóságával, az alakját 45° lefelé irányultsággal és *5px*-nyi életlenítéssel kódolva:

```
box-shadow: 10px 10px 5px; rgba(0,0,0,0.4);
```

A megjelenítés (alul a formázás előtti állapot):



A doboz-árnyék egyik gyakori alkalmazása a gombok formázása. A nyomógomboknak a lekerekített szegélyeknél elkezdett kialakítása a doboz-árnyékkal tovább formázható, amire lépésenkénti példa következik.

A HTML-dokumentumban a <***button***>GOMB<***/button***> tartalmat először a már ismert alakra hozva (lekerekített szegélyek, egyszínű háttér) a CSS-stílus:

```
button { width:160px; height:40px; text-align:center;
    padding:25px; color:#FFEFDF; font-family:Corsiva, serif;
   font-weight:bold; font-size:24px; border:2px solid #EE2E26;
   background-color:#f60; border-radius:20px; }
```

Függőleges belső árnyékot adva a gombnak:

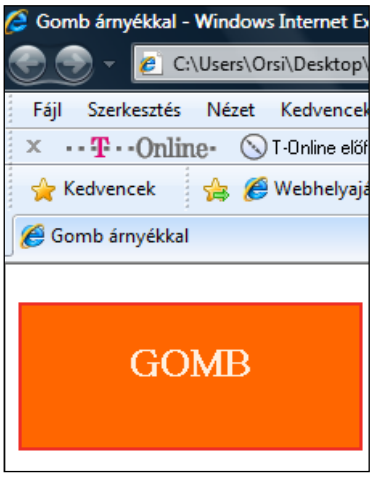
```
button { width:160px; height:40px; text-align:center;
    padding:25px; color:#FFEFDF; font-family:Corsiva, serif;
   font-weight:bold; font-size:24px; border:2px solid #EE2E26;
   background-color:#f60; border-radius:20px; box-shadow:#930
   0px 25px 25px inset; }
```

Fényhatás imitálására egy további árnyékot hozzáadva:

```
button { width:160px; height:40px; text-align:center;
    padding:25px; color:#FFEFDF; font-family:Corsiva, serif;
   font-weight:bold; font-size:24px; border:2px solid #EE2E26;
   background-color:#f60; border-radius:20px; box-shadow:
   rgba(240, 150, 69, 0.5) 0px 15px 2px inset, #930 0px 25px
   25px inset; }
```

A másodszorra felvitt árnyékot az első elé kódoltuk, hogy takarja azt. További belső és külső árnyékok hozzáadásával fokozható a térhatás.

Mindebből az Internet Explorer 8-at használók ezt látják:



A doboz árnyékokhoz felhasznált CSS tulajdonság:

box-shadow doboz-árnyék

3.20. Háttérképek definiálása

A weblap valamennyi elemének (karakter, szó, címsor, sor, bekezdés, lista, táblázat, kép, stb.) van háttérrétege, mely vagy (alapértelmezetten) teljesen átlátszó, vagy a háttérszín (*background-color*) tulajdonság értékének megfelelően kitöltött szín, vagy/és a háttérkép(ek) (*background-image*) tulajdonság értéke(i)ként megadott kép(ek) alkotják.

A háttérképek kódolásának lehetőségeit egy *50px* vastag átlátszó kék (*rgba(0, 0, 255, 0.3)*) szegélyű, *50px* belső margójú, *400px* x *300px* tartalomterületű általános elemen (*div*) mutatjuk be – a tartalom helyzetét a HÁTTÉR beírás jelzi.

Tekintettel arra, hogy mindig javasolt egy jól megválasztott háttérszín megadása is arra az esetre, ha a háttérkép valamilyen ok miatt nem jelenne meg – hogy a tartalom ilyenkor is jól látható legyen - egy sárga *(#FF9)* hátteret is adunk a *div*-nek. Egyúttal a háttér későbbi kódolása során így majd az is kiderül, hogy a háttérszín elhelyezésére is vannak választási lehetőségek.

HTML:

```
<div>HÁTTÉR</div>
```

CSS:

```
div { width: 400px; height: 300px; border:rgba(0, 0, 255, 0.3)
   solid 50px; font-size:36px; font-weight:bold; padding:50px;
   background-color:#FF9; }
```

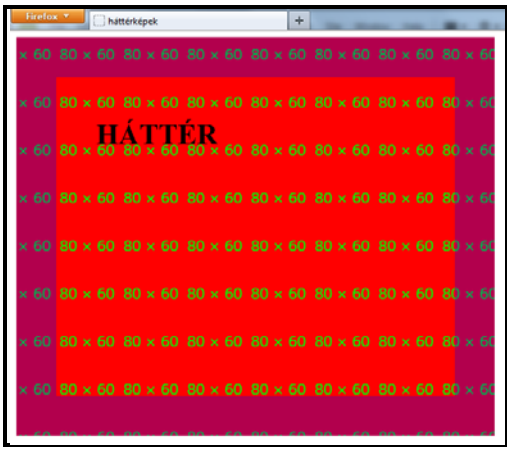
A továbbiakban a fenti HTML/CSS kódolás változatlan marad, és csak a CSS-hez írunk kiegészítéseket. A hátterekkel ellátandó általános elem megjelenítése tehát:

Háttérkép(ek) a **background-image** (háttérkép) tulajdonsággal adható(k) meg, érték az *url* szó és utána szóköz nélkül és zárójelben az előzetesen a gyűjtőmappánkba helyezett háttérkép fájlneve és a fájl kiterjesztése. Először csak egy háttérképpel dolgozunk, így a kód *background-image:url(fájlnév.kiterjesztés)*, pl. egy 80x60-as GIF kitöltőkép esetén (lásd az alábbi képet) *background-image:url(00ff00.gif).*

A kezdő CSS kódhoz tehát hozzáírandó:

```
div { background-image:url(00ff00.gif); }
```

Alapértelmezetten a háttérkép az eredeti méretben vízszintes és függőleges irányban folyamatosan ismétlődik. A belső margó-doboz bal felső sarkához igazodva indul, a szegély külső széleiig (a szegély-dobozt teljesen kitapétázva) tart, tört háttérképek lehetnek a széleken, ha úgy jön ki a méret. A háttérszínt lefedi, a szegély viszont takarja a háttérképet:



3.20.1. Háttérképek ismétlődése

A **background-repeat** (háttérkép ismétlése) tulajdonsággal szabályozható a háttérképek ismétlődése, lehetséges értékei:

- *repeat* (alapértelmezett): a teljes kitöltendő hátteret a kép méretének változatlanul tartásával, annak vízszintes és függőleges irányú ismétlésével összefüggően kitapétázza (lásd fent)
- *repeat-x*: a háttérkép csak vízszintes irányú (x-tengelyű) ismétlését definiálja
- *repeat-y*: a háttérkép csak függőleges irányú (y-tengelyű) ismétlését definiálja
- *no-repeat*: a háttérkép csak egyszeri, ismétlés nélküli megjelenítését definiálja
- *round:* ha a háttérkép egésszámú többszöröse nem pontosan tölti ki a hátteret, a háttérkép úgy méreteződik át, hogy egésszámú többszöröse illeszkedjen a helykitöltéshez
- *space:* az első és utolsó kép a háttér széleihez illeszkedik, a többi háttérkép változatlan méretben olyan térközzel helyezkedik el közöttük, hogy egyenletesen kitöltsék a hátteret

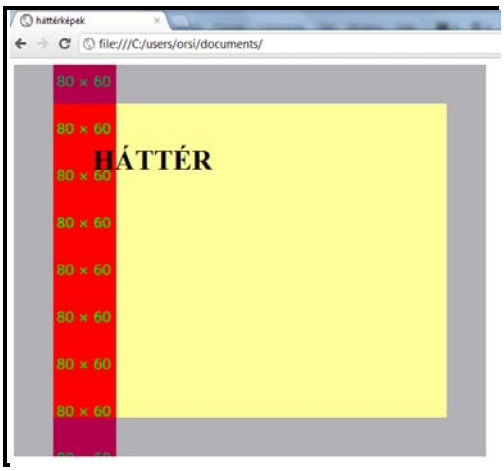
A *repeat-x* érték kódolása és megjelenítése:

```
div { background-image:url(00ff00.gif); background-repeat:
      repeat-x; }
```

A *padding-box* bal felső sarkához igazodva, eredeti méretben, vízszintes irányban a szegélyek külső széleiig folyamatosan tapétáz, a széleknél tört lehet a háttérkép ha úgy jön ki a méret, a háttérszínt fedi, a keret (az átlátszósága mértékében) a háttérképet fedi.

A *repeat-y* érték kódolása és megjelenítése:

```
div { background-image:url(00ff00.gif); background-repeat:
      repeat-y; }
```



A *padding-box* bal felső sarkához igazodva, eredeti méretben, függőleges irányban a szegélyek külső széleiig folyamatosan tapétáz, a széleknél tört lehet a háttérkép ha úgy jön ki a méret, a háttérszínt fedi, a keret (az átlátszósága mértékében) a háttérképet fedi.

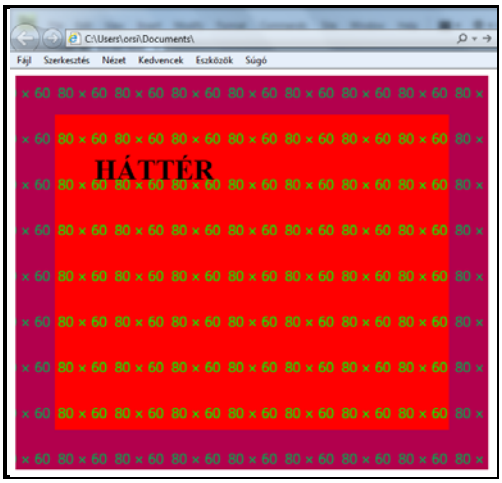
A *no-repeat* érték kódolása és megjelenítése:

```
div { background-image:url(00ff00.gif); background-repeat: no-
      repeat; }
```

A *padding-box* bal felső sarkához igazodva, eredeti méretben, egyszer jelenik meg a háttérszínt fedve a háttérkép:

A *round* érték kódolása és megjelenítése:

```
div { background-image:url(00ff00.gif); background-repeat:
      round; }
```



A háttérkép a belső margó-doboz bal felső sarkához igazodva indul, és mind függőlegesen mind vízszintesen folyamatosan ismétlődik úgy módosítva a háttérkép méretét, hogy a *padding-box* belső élei között egésszámú, egyforma méretű teljes háttérkép legyen. A háttérszínt fedi a háttérkép, a szegély takarja azt.

Figyelem! A *round* értéket az Internet Explorer 8 és a Firefox-ok figyelmen kívül hagyják, *repeat*-ként jelenítik meg.

A *space* érték kódolása és megjelenítése:

```
div { background-image:url(00ff00.gif); background-repeat:
      space; }
```

A háttérkép az eredeti méretben vízszintes és függőleges irányban folyamatosan ismétlődik. A belső margó-doboz bal felső sarkához igazodva indul, mind függőlegesen mind vízszintesen térközöket iktat be úgy, hogy a *padding-box* belső élei között egésszámú, teljes háttérkép legyen. A háttérszín a térközökben kilátszik, a szegély takarja a háttérképet.

Figyelem! A *space* értéket az IE 8 és Firefox nem értelmezik, *repeat*-ként jelenítik meg.

A **background-repeat** tulajdonsággal az *x*- és *y*-irányú ismétlődések egymástól eltérő jelleggel is definiálhatók. A *background-repeat: x-irányú ismétlődés / szóköz / y-irányú ismétlődés* formával az ismertetett értékek kombinációiként változatos tapétázási alakzatok kódolhatók.

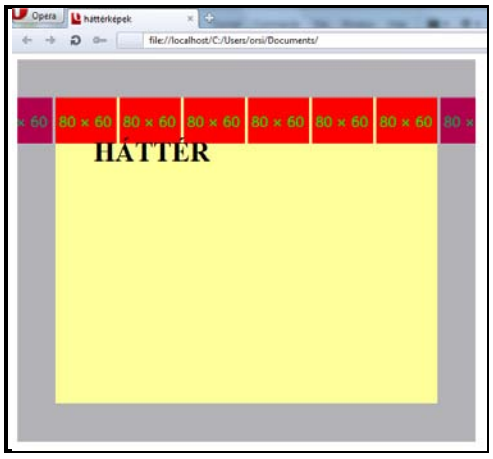
Egyben az eddig használt értékek is felfoghatók összevont alakoknak, amikor a vízszintes és függőleges ismétlés jellege megegyezett:

```
background-repeat:repeat-x repeat-y;
background-repeat:repeat;
```
rövid alakja tehát a

```
background-repeat:no-repeat no-repeat;
background-repeat:no-repeat;
```
rövid alakja tehát a

Egy vízszintes sorban egésszámú, eredeti méretű háttérkép elhelyezésének kódolása például két eltérő érték használatával:

```
div { background-image:url(00ff00.gif); background-repeat:
      space no-repeat; }
```



Megjegyzés: A Firefox-ok és IE 8 esetében a *round*-ra és *space*-re tett korlátozás ennél a kódolási formánál is érvényes (vagyis az alapértelmezett *repeat*-ként jelenítik meg).

3.20.2. Hátterek hatóköre

A háttérkép (és háttérszín!) hatóköre a **background-clip** tulajdonsággal szabályozható, lehetséges értékei:

- *border-box*: ez az alapértelmezett érték; a szegélyek külső élei által határolt teljes területre vonatkozik a háttérszínre és háttérképre megadott érték
- *padding-box*: a belső margó-dobozra vonatkozik a háttérszínre és háttérképre megadott érték
- *content-box*: a tartalom-dobozra vonatkozik a háttérszínre és háttérképre megadott érték

Az összes korábbi kódolás és megjelenítés az alapértelmezett (*border-box*) esetre vonatkozott, úgyhogy most a másik két lehetőségre következik példa.

A *padding-box* érték esetében a kódolás és a megjelenítés:

```
div { background-image:url(00ff00.gif); background-repeat:
      repeat; background-clip: padding-box; }
```

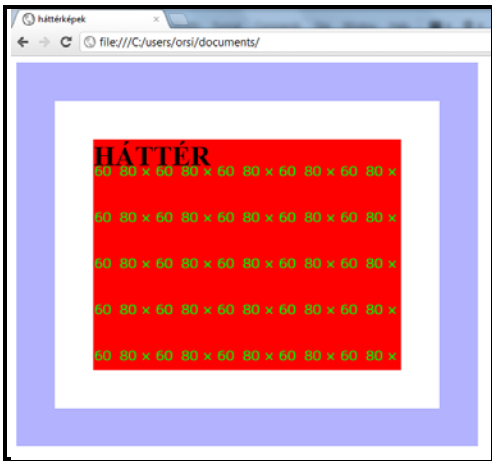
Mind a háttérszín, mind a háttérkép a belső margó-dobozt töltik ki – a részben átlátszó szegély visszanyerte világoskék színét, mert nem tűnik át rajta a sárga háttérszín ill. a piros háttérkép a felirattal.

A *content-box* érték esetében a kódolás és a megjelenítés:

```
div { background-image:url(00ff00.gif); background-repeat:
      repeat; background-clip: content-box; }
```

Mind a háttérszín, mind a háttérkép a tartalom-dobozt töltik ki – a részben átlátszó szegély világoskék színű, mert nem tűnik át rajta a sárga háttérszín ill. a piros háttérkép a felirattal, a belső margó területe színtelen (átlátszó) és nincsen háttérkép sem benne. A tartalom-doboz bal felső sarkát a HÁTTÉR szöveg jelöli.

Figyelem! Az Internet Explorer 8 nem értelmezi a *background-clip* tulajdonságot, kódolása nélkül pedig mindent a szegélydoboz területére helyez el.



3.20.3. Háttérképek helyzete

A háttérkép helyzetének megadásához a pozícionálás referenciapontjának és az ahhoz képest meghatározott helyzetnek az ismerete szükséges.

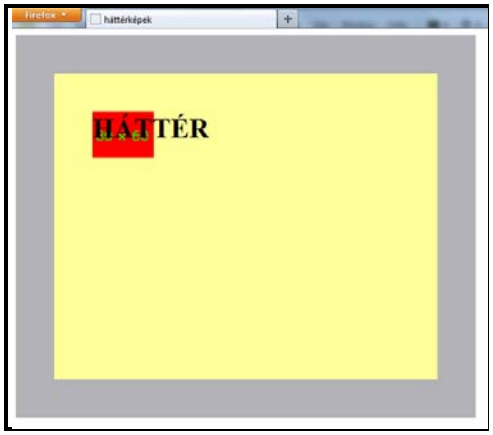
A pozícionálás referenciapontját a **background-origin** tulajdonsággal lehet megadni, melynek lehetséges értékei:

- *padding-box:* ez az alapértelmezett érték; a belső margó-doboz bal felső sarka a helyzetmegadás origója (lásd az összes korábbi példát)
- *content-box:* a tartalom-doboz bal felső sarka a helyzetmegadás viszonyítási pontja
- *border-box:* a szegély-doboz bal felső sarka a pozícionálás referenciapontja

A *background-origin* tulajdonság *content-box* értéke esetében a kódolás és a megjelenítés:

```
div { background-image:url(00ff00.gif); background-repeat: no-
      repeat; background-origin: content-box; }
```

A háttérkép ismétlődés nélkül, eredeti méretben, a tartalom-doboznak a szöveg által jelzett bal felső sarkában helyezkedik el.



A *background-origin* tulajdonság *border-box* értéke esetében a kódolás és a megjelenítés:

```
div { background-image:url(00ff00.gif); background-repeat: no-repeat; background-origin: border-box; }
```

A háttérkép ismétlődés nélkül, eredeti méretben, a szegély-doboz bal felső sarkában helyezkedik el.

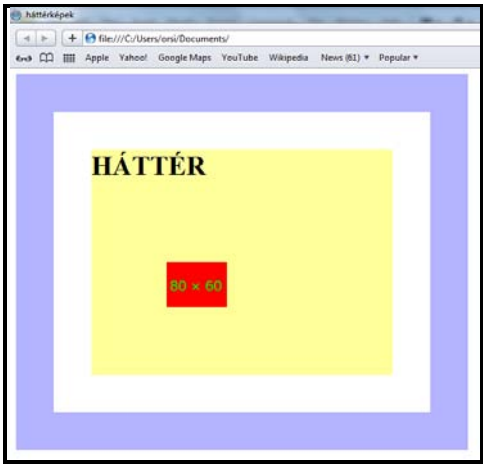
Figyelem! Az Internet Explorer 8 nem értelmezi a *background-origin* tulajdonságot, kódolása nélkül pedig mindíg a belső margó-doboz bal felső sarkát tekinti a helyzetmeghatározás origójának.

A referenciapont ismeretében a **background-position** (háttérkép helyzete) tulajdonság révén a háttérkép pozíciója most már egyértelműen definiálható. A *background-position* megadható névvel (*keyword*), *px*-ben vagy százalékosan.

Névvel a *left* (balra), *right* (jobbra), *top* (felülre) *bottom* (alulra), és *center* (középre) pozíciók definiálhatók – először a vízszintes (*left, center, right*), majd a függőleges (*top, center, bottom*) helyzetet kell megadni. A *background-position* tulajdonság alapértelmezett értéke *0% 0%*, vagy *0px 0px* vagy *left top*. A 100% 100% a jobb alsó sarok (*right bottom*), az 50% 50% a közép (*center center)*, Az 50% vagy *center* önmagában is a közepet jelenti, mert ha a pozícióra csak egy érték van megadva, a második értéket automatikusan *center*-ként értelmezik a böngészők. Negatív értékek is megadhatók a háttérkép helyzetére, ekkor az adott elem hátterének széléhez képest kifelé pozícionálódik a háttérkép.

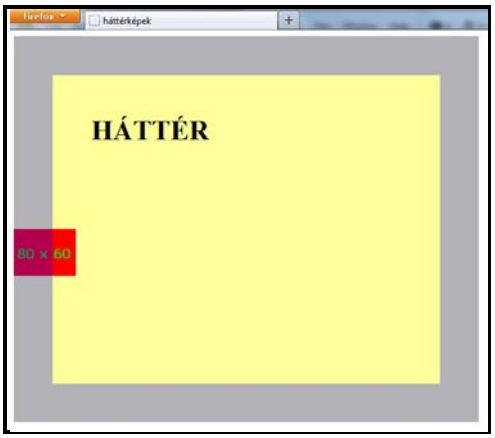
A tartalom-doboz sarkához képest vízszintesen 100px-re, függőlegesen 150px-re helyezett háttérkép és csak a tartalom-dobozt kitöltő háttérszín kódolása és megjelenítése:

```
div { background-image:url(00ff00.gif); background-repeat: no-repeat; background-clip: content-box; background-origin: content-box; background-position:100px 150px; }
```



Megjegyzés: Ha a vízszintes pozíciót *-100px*-re változtatnánk, a háttérkép kívül kerülne a tartalom-dobozon – és mivel a háttér hatóköre a tartalomdobozra van beállítva, ezért egyáltalán nem látszódna. Ha a háttér hatókörét kiterjesztjük a belső margó-dobozra, a háttérkép részben – a margó-doboz éléig – válik láthatóvá, a szegély-dobozra kiterjesztett háttérnél viszont a szegélyen is áttűnve teljes méretben megjelenik:

```
div { background-image:url(00ff00.gif); background-repeat: no-
      repeat; background-clip: border-box; background-origin:
      content-box; background-position:-100px 150px; }
```



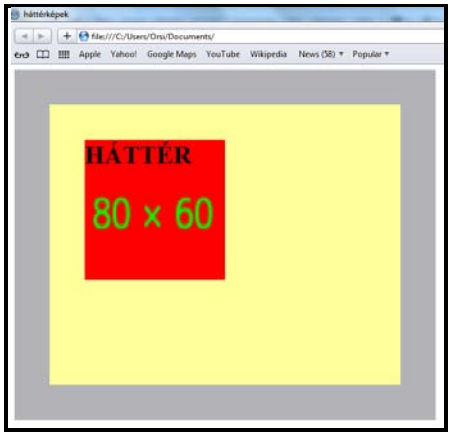
3.20.4. Háttérképek mérete

A **background-size** tulajdonsággal a háttérkép méretét lehet változtatni. Lehetséges értékei: *auto*, szélesség és magasság abszolút mérete pixelben, szélesség és magasság a kitöltendő rész százalékában, *cover* (lefed), *contain* (magában foglal). Figyelni kell arra, okoz-e pixelesedést a háttérkép felnagyítása!

Az átméretezéskor a háttérkép akkor nem torzul, ha a szélességét (pixelben vagy 100%-ban) megadjuk, a magasságára pedig *auto*-t definiálunk – ekkor a magasságot az eredeti képarány szerint határozza meg a böngésző. Ha csak egy érték van megadva, a másikat *auto* – nak értelmezik a böngészők. Az *auto auto* vagy egyszerűen *auto* a méretek változatlanul tartását jelenti – ez az alapértelmezett érték, az eddigi példák mind erre vonatkoztak.

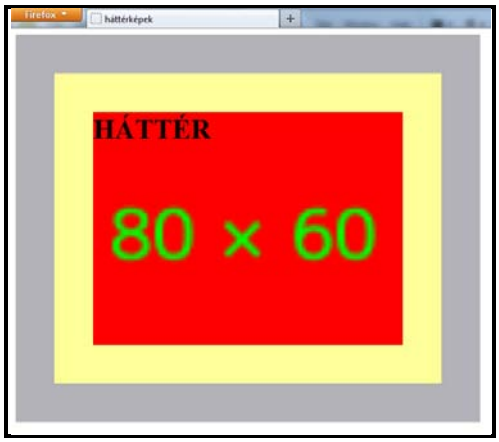
Háttérképünket *200px* x *200px*-re átméretezve (mely nem csak méret növekedést, hanem méretarány torzulást is jelent) a kódolás és megjelenítés:

```
div { background-image:url(00ff00.gif); background-repeat: no-
      repeat; background-origin: content-box; background-size:
      200px 200px; }
```



Példa a *contain* érték kódolására és megjelenítésére:

```
div { background-image:url(00ff00.gif); background-repeat: no-
      repeat; background-origin: content-box; background-size:
      contain; }
```



A *contain* értéknél változatlan képarány mellett a háttérkép mérete úgy változik, hogy a teljes háttérkép beleférjen a háttérként kijelölt területbe.

A *cover* értéknél változatlan képarány mellett a háttérkép mérete úgy változik, hogy a legkisebb mérete teljesen lefedje a háttérként kijelölt területet.

Figyelem! Az Internet Explorer 8 nem veszi figyelembe a *background-size* tulajdonságot, tehát változatlan háttérkép mérettel jeleníti meg a hátteret.

3.20.5. Háttérképek görgethetősége

A **background-attachment** (háttérkép csatolása) tulajdonsággal adható meg a háttérképnek a különböző viszonyítási alapon előírt görgethetősége. Lehetséges értékei:

- *scroll* (görgetett): ez az alapértelmezett érték; amennyiben a dokumentum mérete nagyobb a kijelző méreténél, és ezért megtekintéséhez görgetésre van szükség, a háttér a dukumentum görgetősávjával – azaz a tartalommal - együtt mozog
- *local* (helyi): a háttér egy elem tartalmához rögzített – amennyiben az elem nem fér el a tárolójában és ezért görgetni kell, az így definiált háttér az elem tartalmával együtt mozog
- *fixed* (rögzített): a háttér a dokumentum görgetésétől függetlenül állandó helyzetben marad

A *local* érték kódolásához és megjelenítéséhez egy későbbi fejezetben ismertetett tulajdonság, az *overflow* (túlnyúlás) felhasználását is igénybe kell venni, a HTML-kódban pedig sok függőleges sort kell írni a tartalomba, hogy tényleg szükség legyen a *div* elem (nem a dokumentum!) görgetősávjára.

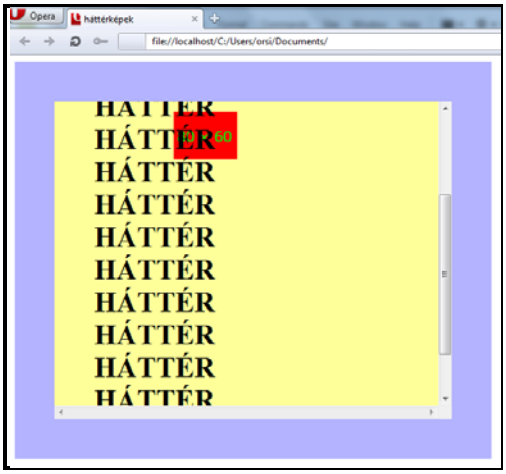
Az ezt megvalósító CSS kódolás:

```
div { background-image:url(00ff00.gif); background-repeat: no-
      repeat; background-clip: padding-box; background-origin:
      content-box; background-position:100px 150px;
      overflow:scroll; background-attachment: local; }
```

A HTML-kódban a sok függőleges sor a görgetősáv kedvéért:

```
<div>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR
<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR
<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR<br>HÁTTÉR
</div>
```

A megjelenítés:



A jobb oldali függőleges görgővel – a *div* tartalmával - együtt fog mozogni a háttérkép.

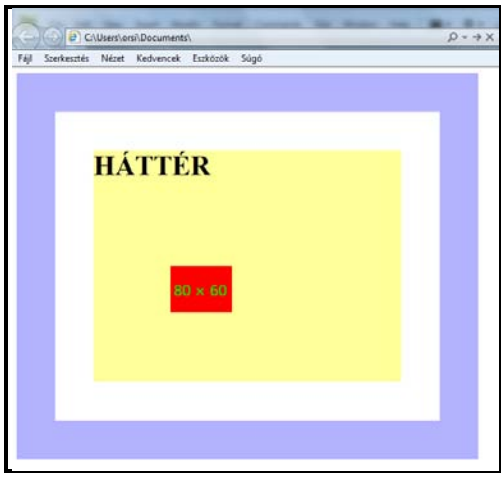
3.20.6. Összevontan megadható háttérkép tulajdonságok

A háttérkép tulajdonságok **background**-ként összevonva (*shorthand*) is megadhatók, a tulajdonságok sorrendje kötött, az értékek vessző nélküli szóközzel vannak elválasztva egymástól:

background: background-image background-position / background-size background-repeat background-attachment background-clip background-origin background-color;

Konkrét példa egy ismétlődés nélküli, eredeti méretű, a dokumentum tartalmával együtt görgethető, háttérszínnel is rendelkező, a tartalom-doboz háttereként szolgáló, a tartalom-dobozhoz viszonyított helyzetű háttérkép összevont kódolására:

```
background:url(00ff00.gif) 100px 150px / auto no-repeat scroll
content-box content-box #FF9;
```

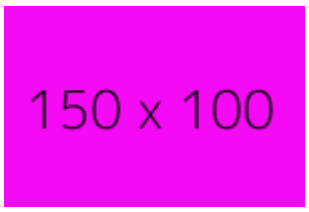


Megjegyzés: Tekintettel az új tulajdonságokra és/vagy értékekre, melyeket a böngészők még nem egyöntetűen támogatnak, az összevont forma könnyen értelmezhetetlenné válhat némelyikük számára, ezért célszerűbb egyelőre a tulajdonságokat külön-külön (*longhand*) definiálni.

3.20.7. Több háttérkép egyidejű alkalmazása

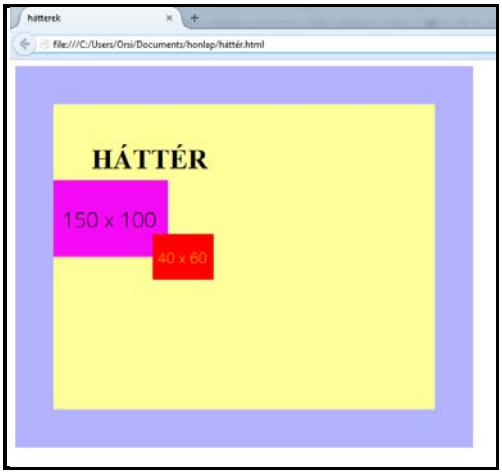
Ha több, egymásra vagy egymás mellé helyezett képből épül fel a háttér, felsorolás-szerűen, vesszővel és szóközzel elválasztva kell a (*.jpg*, *.png* vagy *.gif* kiterjesztésű) kép-fájlokat és az egyes tulajdonságaikat megadni. A képek egymásra rétegződésének sorrendjét a felsorolás sorrendje szabja meg – az első fájlnév képe van legfelül (a felhasználóhoz legközelebb), és minden további kép a sorrendnek megfelelően egyre hátrább.

Vegyünk egy második háttérképet *url(0011ff.gif)* névvel/kiterjesztésssel:



Az eredeti *div*-et most két háttérképpel ellátva a kódolás és a megjelenítés:

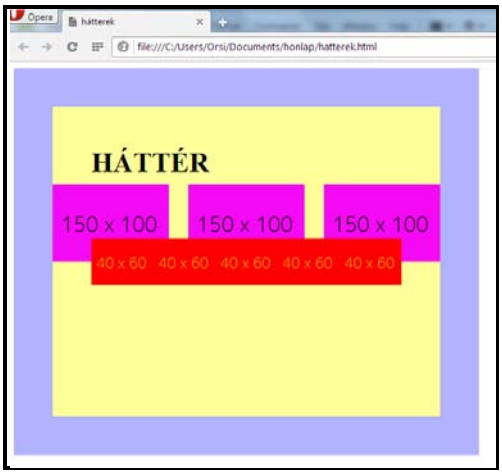
```
div { background-image:url(00ff00.gif), url(0011ff.gif);
      background-repeat:no-repeat, no-repeat;
      background-position: 80px 120px, 0px 100px;
      background-origin: content-box, padding-box;
      background-clip: content-box, padding-box; }
```



Figyelem! Az Internet Explorer 8 nem értelmezi a többszörös háttérképeket.

Az IE9 és az újabb böngészők – a Firefox-oknál korábban említett korlátokkal – további tulajdonság/érték párokat (háttérkép variációkat) is meg tudnak jeleníti, pl.:

```
div { background-image:url(00ff00.gif), url(0011ff.gif);
      background-repeat:round no-repeat, space no-repeat;
      background-position: 80px 120px, 0px 100px;
      background-origin: content-box, padding-box;
      background-clip: content-box, padding-box; }
```



3.20.8. Hátterek keverése

A hátterek - háttérszín és háttérkép(ek) – keverésével (korábban csak képszerkesztő programokkal létrehozható) változatos vizuális hatások érhetők el. A *background-blend-mód* tulajdonság lehetséges értékei:

- *normal*: ez az alapértelmezett – semmilyen keverés nem történik, a megszokott módon jelennek meg a háttérképek
- *multiply* (szendvicspozitív): részben áttetszővé és sötétebbé teszi az adott háttérképet megtartva a kontrasztot
- *screen* (szendvicsnegatív): részben áttetszővé és világosabbá teszi az adott háttérképet megtartva a kontrasztot
- *overlay* (átfedés): az alatta elhelyezkedő hátterek színének függvényében a szendvicspozitív vagy szendvicsnegatív keverési módot alkalmazza
- *darken* (sötétítés): az adott és az alatta lévő háttér képpontok színe közül a sötétebbet mutatja
- *lighten* (világosítás): az adott és az alatta lévő háttér képpontok színe közül a világosabbat mutatja
- *color-dodge* (színfakítás): részben áttetszővé és világosabbá teszi az adott háttérképet, és részben megtartja a kontrasztot
- *color-burn* (színégetés): részben áttetszővé és sötétebbé teszi az adott háttérképet, és részben megtartja a kontrasztot
- *hard-light* (kemény fény): növeli a kontrasztot és részben áttetszővé teszi az adott hátteret
- *soft-light* (lágy fény): növeli a kontrasztot és részben áttetszővé teszi az adott hátteret
- *difference* (különbség): megváltoztatja az adott háttérkép színeit, miközben a kontraszt nagy részét megtartja
- *exclusion* (kivétel): részlegesen megváltoztatja az adott háttérkép színeit, de nem sokat tart meg a kontrasztból
- *hue* (színezet): az adott háttérkép színezetét használja az alsó háttér fényességével és telítettségével
- *saturation* (telítettség): az adott háttérkép telítettségét használja az alsó háttér fényességével és színezetével
- *color* (szín): az adott háttérkép színét használja az alsó háttér fényességével
- *luminosity* (fényesség): az adott háttérkép fényességét használja az alsó háttér színével

A hátterek keverési módjainak hatása nem annyira a definíciók megtanulásával, hanem inkább sok próbálgatással, kísérletezéssel érthető meg. Ezért (a szemléletes demonstrálás céljából) egymás mellé helyezzük az eredeti több háttérképes megjelenítést és a keverési módokat variáló megjelenítést, ahol a különböző keverési kombinációk hatásai közvetlenül érzékelhetők.

HTML:

```
<div id="nemkevert">HÁTTÉR</div>
<div id="kevert">HÁTTÉR</div>
```

CSS:

```
<style>
    div { width: 400px; height: 300px; border:rgba(0, 0, 255,
          0.3) solid 50px; font-size:36px; font-weight:bold;
          padding:50px; background-color:#FF9; display:inline-
          block; background-image:url(00ff00.gif),
          url(0011ff.gif); background-repeat:round no-repeat,
          space no-repeat; background-position: 80px 120px,
          0px 100px; background-origin: content-box, padding-
          box; background-clip: content-box, padding-box; }
</style>
```

Ez a formázás megegyezik a korábbival, csak a két *div* egymás mellé helyezése érdekében bekerült a *display:inline-block* tulajdonság/érték:

A jobb oldali *div* hátterének különböző keverési módjait a CSS-kódba beírt különböző értékekkel lehet kitapasztalni – az egyes háttérképekre vonatkozó értékeket vesszővel elválasztva, felsorolásként lehet megadni (mint a többes háttérképeknél):

```
#kevert { background-blend-mode: ........, .........; }
```

Ha pl. az első hátterünk keverési módjára *kivétel*-t, a másodikéra *szendvicspozitív* értéket definiálunk, akkor:

```
#kevert { background-blend-mode: exclusion, multiply; }
```

Megjegyzés: Az Internet Explorer-ek egyike sem, a Safari csak a 8.0-tól értelmezi a *background-blend-mode* tulajdonságot.

<u>A háttérképekhez felhasznált CSS tulajdonságok:</u>

background-image	háttérkép(ek) definiálása
background-repeat	háttérkép(ek) ismétlődése
background-clip	háttér hatóköre
background-origin	háttérkép(ek) referencia pontja
background-position	háttérkép helyzete
background-size	háttérkép(ek) mérete
background-attachment	háttér csatolása a tartalomhoz
background-blend-mode	háttérkép(ek) keverése
background	összevont háttér tulajdonságok

3.21. Táblázatok formázása

Egy táblázat (*table*) opcionális címből (*caption*) és belső táblázatból (*internal table* – ilyen címke nincsen, ez a szűkebb értelemben vett táblázatot jelenti) – azaz sorokba (*table row*) rendezett cellákból (*table header*, *table data*) áll.

A belső (szűk értelemben vett) táblázat szerkezetének esetleges további tagolását és formázásának kialakítását a táblázatfej (*table head*), táblázattörzs (*table body*), táblázat lábléc (*table footer*), táblázat oszlop (*col*) és táblázat oszlopcsoport (*colgroup*) címkék segítik.

A különböző formázási lehetőségeket az alábbi táblázaton mutatjuk be:

Ízelítő az angolszász hosszmértékekből			
angol elnevezés	***magyar fordítás***	***váltószám***	***metrikusan***
inch	hüvelyk	=12* line	=2,54 cm
foot	láb	=12 inch	=30,48 cm
yard	rőf	=3 foot	=91,44 cm
fathom	öl	=2 yard	=1,83 m
pole	rúd	=5½ yard	=5,03 m
furlong	?	=40 pole	=201,16 m
statute mile	angol/szárazföldi mérföld	=8 furlong	=1609,33 m

*egyes források 10-es váltószámot adnak meg

A táblázat HTML-kódolása:

```
<table>
    <caption>Ízelítő az angolszász hosszmértékekből</caption>
    <thead>
        <tr><th>angol elnevezés</th><th>magyar fordítás</th>
            <th>váltószám</th><th>metrikusan</th></tr>
    </thead>
    <tbody>
        <tr><th>inch</th><td>hüvelyk</td><td>=12* line</td>
            <td>=2,54 cm</td></tr>
        <tr><th>foot</th><td>láb</td><td>=12 inch</td>
            <td>=30,48 cm</td></tr>
        <tr><th>yard</th><td>rőf</td><td>=3 foot</td>
            <td>=91,44 cm</td></tr>
        <tr><th>fathom</th><td>öl</td><td>=2 yard</td>
            <td>=1,83 m</td></tr>
        <tr><th>pole</th><td>rúd</td><td>=5½ yard</td>
            <td>=5,03 m</td></tr>
        <tr><th>furlong</th><td>?</td><td>=40 pole</td>
            <td>=201,16 m</td></tr>
        <tr><th>statute mile</th><td>szárazföldi mérföld</td>
            <td>=8 furlong</td><td>=1609,33 m</td></tr>
     </tbody>
     <tfoot>
        <tr><td colspan="4"> *egyes források 10-es váltószámot
             adnak meg</td></tr>
     </tfoot>
</table>
```

A *thead*, *tbody* és *tfoot* alkalmazása nem szükségszerű, a példa kedvéért vettük be.

Az alapértelmezett – formázatlan, a HTML-részből jólismert – megjelenítés:

Firefox táblázatok

Ízelítő az angolszász hosszmértékekből

angol elnevezés	magyar fordítás	váltószám	metrikusan
inch	hüvelyk	=12* line	=2,54 cm
foot	láb	=12 inch	=30,48 cm
yard	rőf	=3 foot	=91,44 cm
fathom	öl	=2 yard	=1,83 m
pole	rúd	=5½ yard	=5,03 m
furlong	?	=40 pole	=201,16 m
statute mile	szárazföldi mérföld	=8 furlong	=1609,33 m

*egyes források 10-es váltószámot adnak meg

3.21.1. Nem táblázat-specifikus tulajdonságok alkalmazása

Az előző CSS-fejezetekben ismertetett tulajdonságok – azon néhány eset kivételével, melyeket kifejezetten megnevez a szabvány – a táblázatok formázására is felhasználhatók, bár a hatásuk kifejtésének módja részben eltérő lehet.

Kódoljunk a sablonnak választott *table* táblázat-címkének külső és belső margót, hátteret, szegélyt, betűszínt, betűméretet és szövegigazítást:

```
table { margin-left:150px; background-color:#FF0; background-
        clip:content-box; border: 2px solid blue; border-radius:
        10px; padding:25px; text-align:right; color:#00F; font-
        size: 24px; }
```

A fenti formázás megjelenítése:

Firefox táblázatok

Ízelítő az angolszász hosszmértékekből

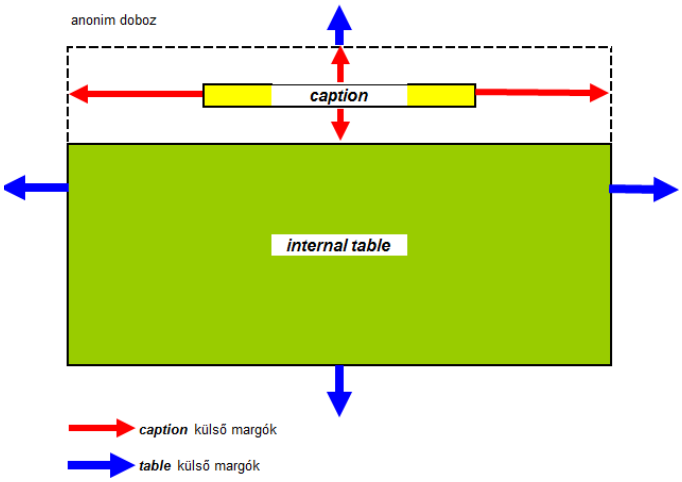
angol elnevezés	magyar fordítás	váltószám	metrikusan
inch	hüvelyk	=12* line	=2,54 cm
foot	láb	=12 inch	=30,48 cm
yard	rőf	=3 foot	=91,44 cm
fathom	öl	=2 yard	=1,83 m
pole	rúd	=5½ yard	=5,03 m
furlong	?	=40 pole	=201,16 m
statute mile	szárazföldi mérföld	=8 furlong	=1609,33 m

*egyes források 10-es váltószámot adnak meg

A kódolás és megjelenítés összevetéséből megállapítható, hogy:

- a *margin* külső margó tulajdonsággal a teljes (cím + belső) táblázat helyzetét lehet pozícionálni
- a *padding* belső margó, *border* szegély, *background* háttér, *és text-align* szövegigazítás (tehát a helyzet kivételével valamennyi, valamilyen dobozhoz köthető) tulajdonságokat csak a belső táblázatra értelmezik a böngészők
- a dobozoktól független formázások (szín, betűméret) a teljes (cím + belső) táblázatra értelmezettek

A táblázat szerkezete formázási szempontból tehát:



A táblázat címke (*table*) generál egy „*anonim dobozt*”, mely a szorosan értelmezett táblázat dobozt (*internal table* = belső táblázat) és a cím-dobozt (*caption*) foglalja magában. Az „anonim doboz” alkalmas a teljes táblázat pozícionálására, azonban nem hivatkozható, a cím és a szorosan értelmezett táblázat együttes formázását egyéb módon kell megvalósítani.

a) A cím és belső táblázat közös háttérrel, szegéllyel és belső margóval való formázása a táblázat *div*-be foglalásával, és a CSS-kódolásnak az így kapott *div*-hez rendelésével oldható meg (és az egyszerűség kedvéért ehhez a külső margó is hozzáírható):

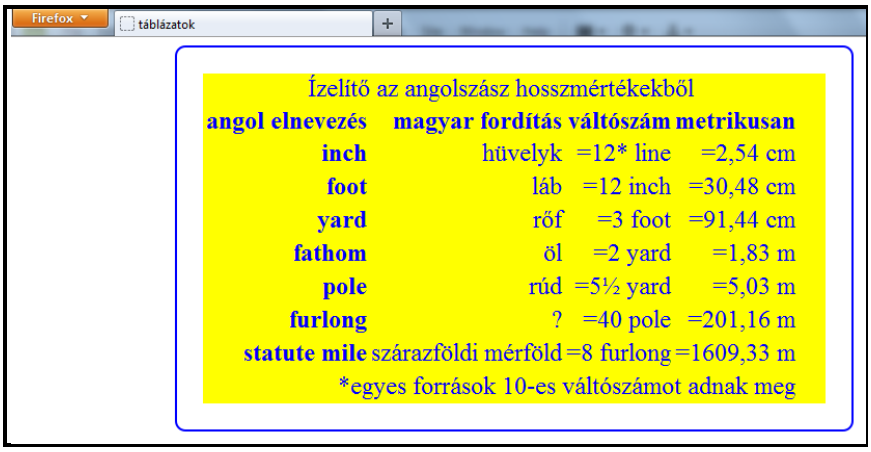
HTML:

```
<div>
   <table>.........................................</table>
</div>
```

CSS:

```
div { margin-left:150px; background-color:#FF0; background-
      clip:content-box; border:2px solid blue; border-radius:
      10px; padding:25px; text-align:right; color:#00F; font-
      size:24px; width:600px; }
```

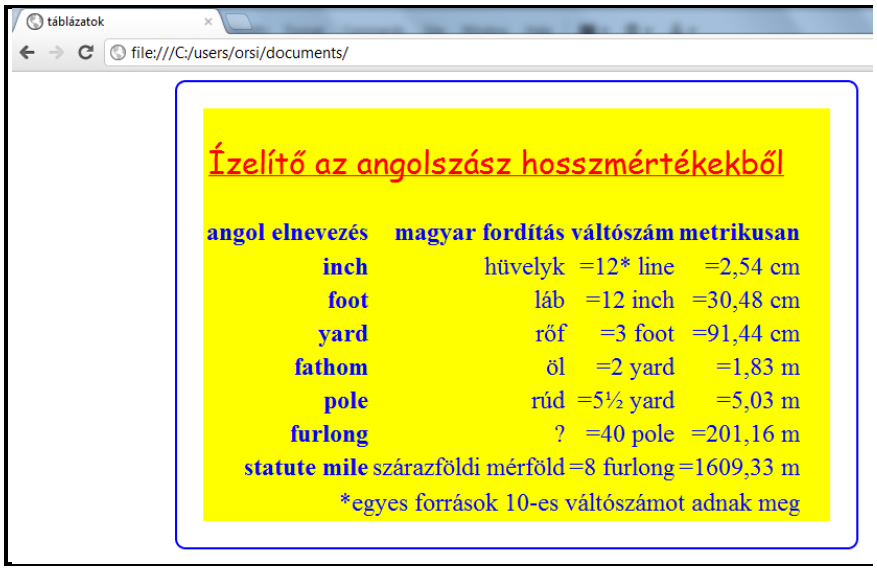
A *width* azért szükséges, hogy a háttérnek és keretnek a vízszintes kiterjedését korlátozzuk a táblázat vízszintes kiterjedéséhez.



b) A cím formázását egy önálló szövegblokk-ként külön kell kódolni, pl.:

CSS:

```
div { margin-left:150px; background-color:#FF0; background-
      clip:content-box; border:2px solid blue; border-radius:
      10px; padding:25px; text-align:right; color:#00F; font-
      size:24px; width:600px; }

caption { padding:30px 5px; text-align:left; color:#f00; font-
          size:28px; font-family:"Comic Sans MS", cursive; text-
          decoration:underline; }
```



A megjelenítésből látszik, hogy a *table*-hez ill. a *div*-hez megadott tulajdonságok értékeit egymást átfedő formázási utasítások esetén a *caption*-hez rendelt értékek felülírják, tehát a cím szabadon formázható.

Megjegyzés: A *caption* cím külön háttérrel, kerettel, doboz-árnyékkal, szélességgel, stb. is ellátható a fenti kódolást tovább bővítve. Például:

```
caption { background-color:#3F3; border:4px dashed #F3C;
          width:600px; }
```

c) A cím egy kiragadott része a *span* HTML-címke beiktatásával és CSS-tulajdonságok hozzárendelésével, a belső táblázat egyes sorai vagy cellái pedig a HTML-kódban meglévő egyedi vagy *id* jellemzővel azonosított címkéhez rendelt CSS-tulajdonságokkal tovább formázhatók. Például:

HTML:

```
<caption>Ízelítő az <span>angolszász</span> hosszmértékekből
</caption>
```

CSS:

```
span { font-variant:small-caps; font-weight:bolder; color:
       #33C; }
```

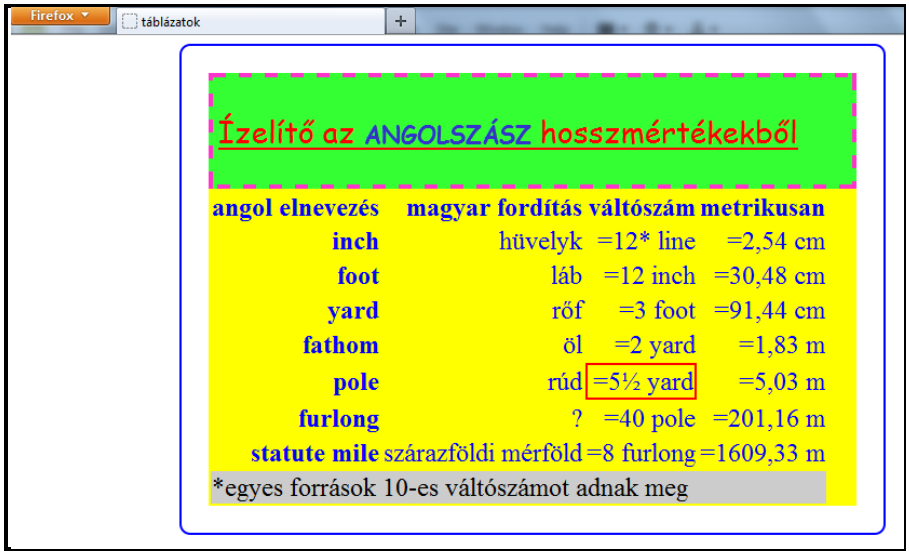
ill. HTML:

```
<tr><th>pole</th><td>rúd</td><td id="pole">=5½ yard</td>
    <td>=5,03 m</td></tr>
```

CSS:

```
#pole { border:2px solid red; }
tfoot { background-color:#CCC; color:black; text-align:left; }
```

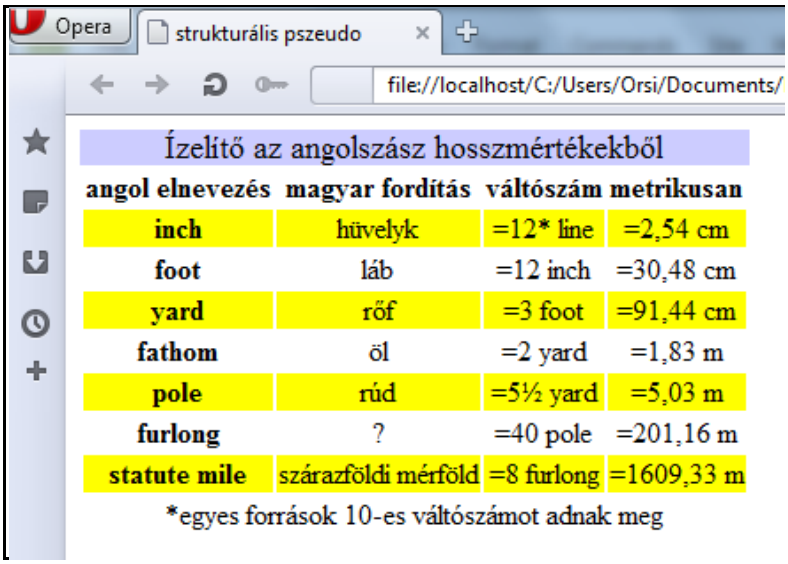
A fenti kódok hozzáadása következtében a formázott megjelenítés:



d) A táblázatok könnyebb áttekinthetősége érdekében gyakran alkalmazzák a „zebracsíkozást”, azaz az egymást követő sorok háttérszínének váltogatását. Ezt korábban a páros és páratlan sorok egy-egy osztályba sorolásával, majd ezeknek az osztályoknak a formázásával állították elő. Ennek hátránya, hogy minden sorhoz be kell kódolni a *class* osztálykijelölőt, és ha később egy sort betoldására vagy törlésére kerül sor, akkor át kell az osztálykijelölést írni.

Strukturális ál-osztályú kijelölés alkalmazásával mindez elkerülhető az alábbiak szerint:

```
<style>
   caption { background-color:#CCF; font-size:18px; }
   table { text-align:center; }
   tbody tr:nth-child(odd) { background-color:yellow; }
</style>
```



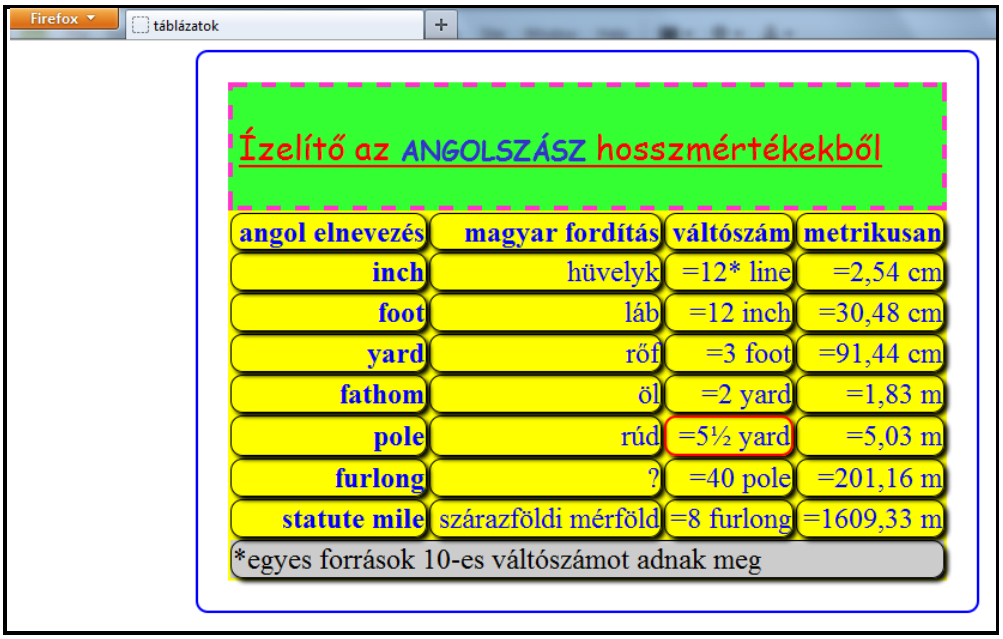
Hasonló módon lehetne pl. csak a harmadik sort kijelölni és formázni (*származtatott kijelölés* helyett a gyakorlás kedvéért *gyermek kombinátort* alkalmazva):

```
<style>
    caption { background-color:#CCF; font-size:18px; }
    table { text-align:center; }
    tbody > tr:nth-child(3) { background-color: yellow; };
</style>
```

A formázás a sorok esetleges átvariálása esetén is változatlanul a mindenkori harmadik soron marad.

e) A belső táblázat celláihoz (*th, td*) a *border* tulajdonság ismert lehetőségeivel kódolható szegély. Például:

```
th, td { border:1px solid black; border-radius:10px; box-
         shadow:3px 3px 3px black; }
```



3.21.2. Táblázat-specifikus tulajdonságok alkalmazása

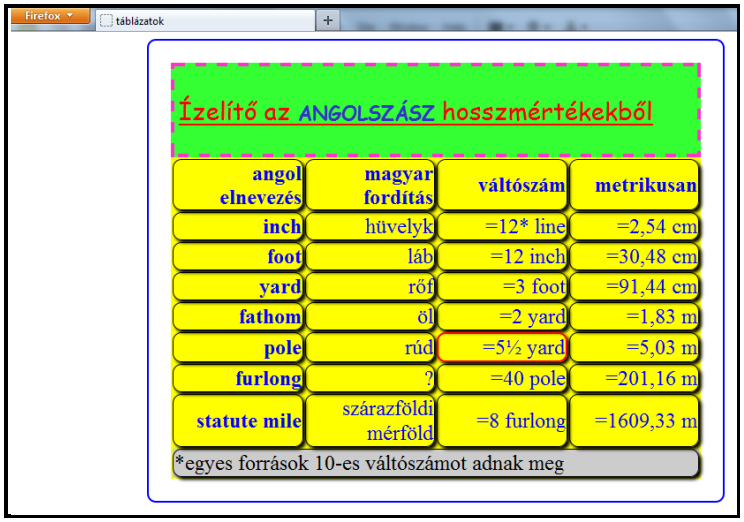
Az általános tulajdonságokon felül vannak csak a táblázatokra jellemző és kódolható tulajdonságok:

a) A táblázat alaprajzát a **table-layout** (táblázat elrendezése) tulajdonsággal lehet meghatározni. Lehetséges értékei:

- *auto:* ez az alapértelmezett érték; a böngésző a cellatartalmak alapján képezi az oszlopszélességeket (az eddigi összes példa az alapértelmezett értéket használta)
- *fixed:*a böngésző egyenlő szélességű oszlopokba helyezi el a cellatartalmakat

A táblázat szélességét feltétlenül meg kell adni, hogy a böngésző tudja, milyen alapon kell számolnia az elrendezést. A *table-layout:fixed* kódolása és megjelenítése:

```
table { table-layout:fixed; width:600px; }
```

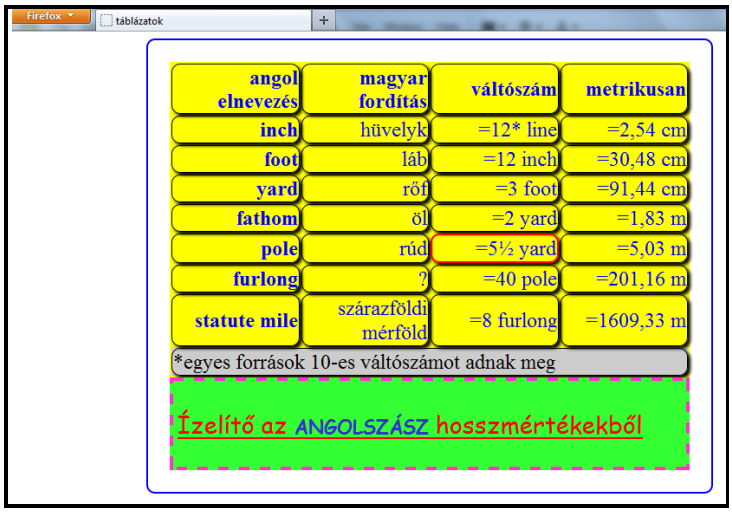


b) A cím elhelyezését definiálja a **caption-side** tulajdonság, melynek lehetséges értékei:

- *top* (*felül*):ez az alapértelmezett; az összes eddigi példa ezt az értéket használta
- *bottom* (*alul*): a táblázat alján jelenik meg a táblázat címe

A *caption-side:bottom* kódolása és megjelenítése:

```
table { caption-side:bottom; }
```



c) A belső táblázat celláinak szegélyeit két, egymástól eltérő modellben lehet definiálni, melyek a **border-collapse** (szegélyek összevonása) tulajdonsággal határozhatók meg. Lehetséges értékei:

- *collapse (összevonva):* főleg akkor használatos, ha a táblázatban az elejétől a végéig folyamatos szegélyeket alkalmazunk

- *separate (elválasztva):* az egyes cellákhoz rendelt szegélyek esetében gyakoribb a használata. A szegélyek közötti távolság a *border-spacing* tulajdonsággal definiálható.

A belső táblázat cellaszegélyeinek *border-collapse:collapse* kódolása és megjelenítése:

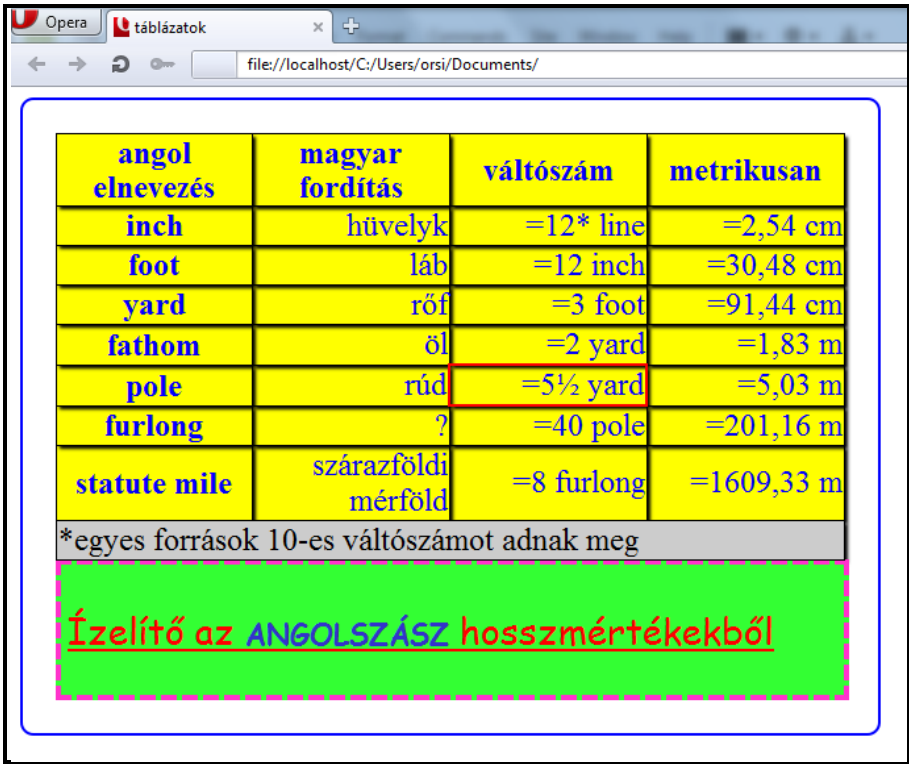
```
table { border-collapse:collapse; }
```

A megjelenítésből látható, hogy

- a cellaösszevonás felülírja a sarkok lekerekítését, hiszen az ellentétes irányú görbületeket nem lehet összeolvasztani

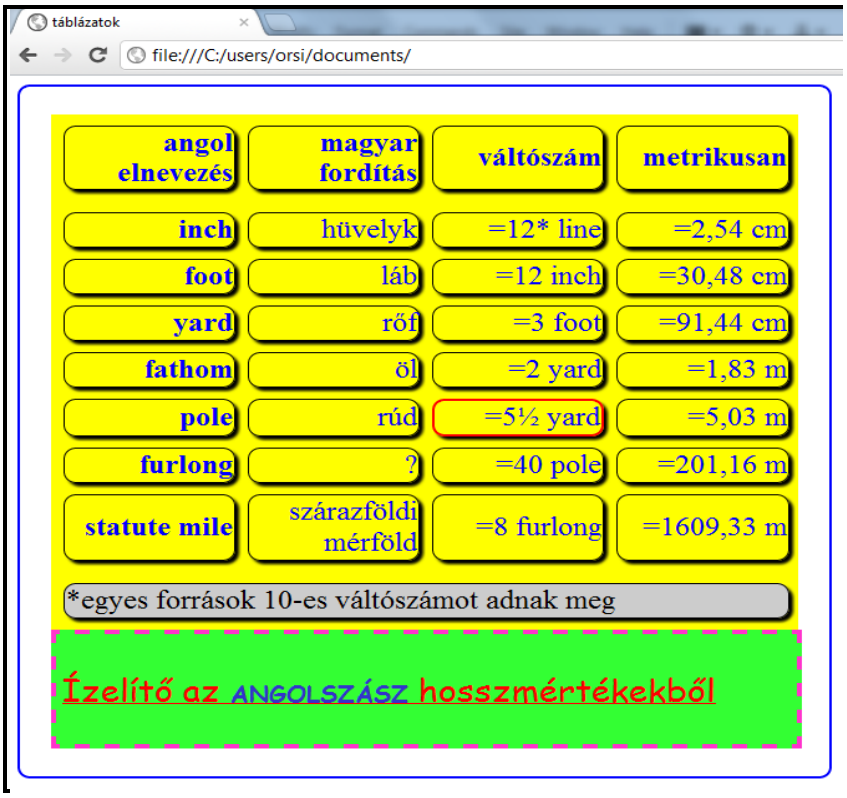
- az eltérő szegélyek egybeesése „szegély kofliktust” okoz, melyet a böngésző a szegélykonfliktus feloldására vonatkozó szabályaival old meg – általában a nagyobb szegélyvastagságnak, látványosabb szegélystílusnak, hasonló jelleg esetén a bal oldalinak vagy fentebb lévőnek van elsőbbsége

- a *border-spacing* esetleges beírása nem okoz semmit, hiszen itt nem értelmezett



A belső táblázat cellaszegélyeinek *border-collapse:separate* és *border-spacing:……..px* kódolása és megjelenítése:

```
table { border-collapse:separate; border-spacing:10px; }
```



d) A belső táblázat üres celláinak megjelenítése az **empty-cells** tulajdonsággal határozható meg, lehetséges értékei:

- *show* (*mutatja*): ez az alapértelmezett érték; a cellát minden formázásával együtt, de üres tartalommal mutatja

- *hide* (*elrejti*):az üres cella helyén nem jelenik meg semmi, mintha egyáltalán nem is létezne

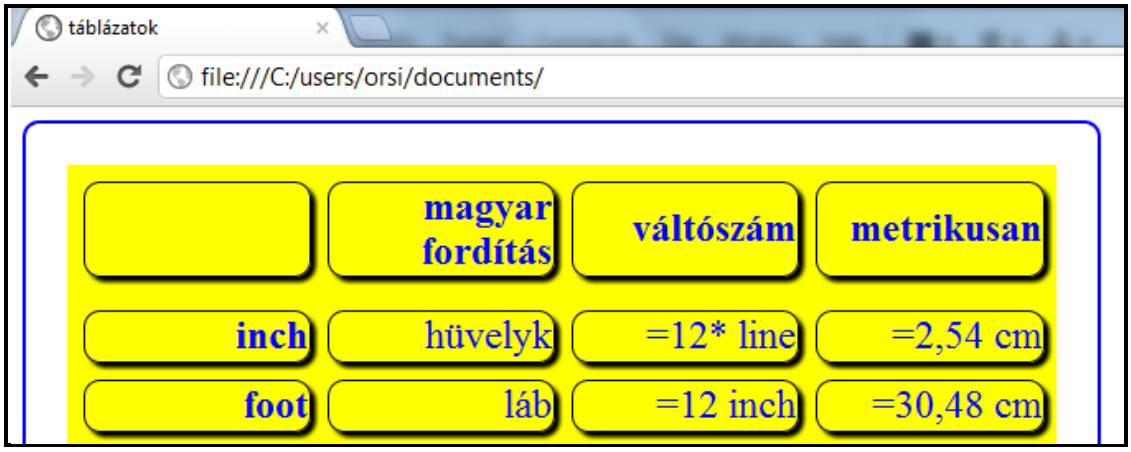
Az „angol elnevezés” cellából kivesszük a szöveget, hogy legyen üres cellánk:

HTML:

```
<thead>
    <tr><th></th><th>magyar fordítás</th><th>változám</th>
        <th>metrikusan</th></tr>
</thead>
```

CSS:

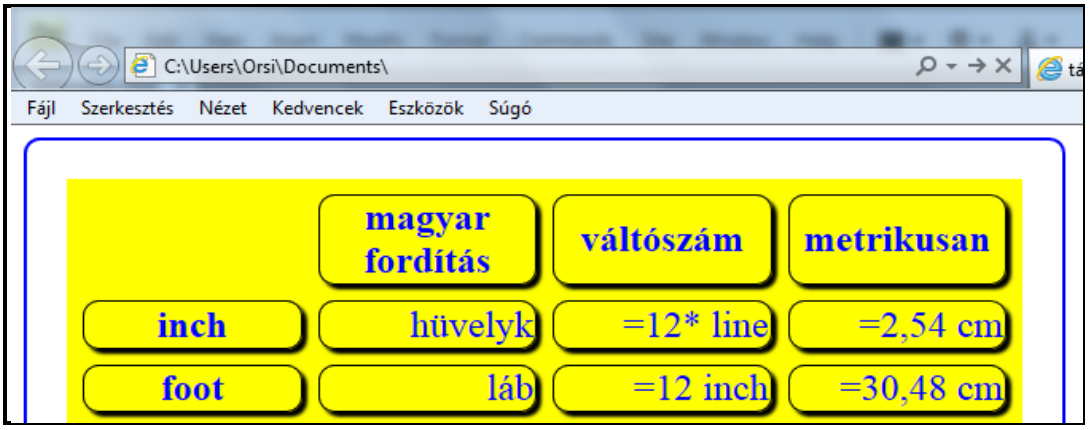
```
table { empty-cells:show; }
```



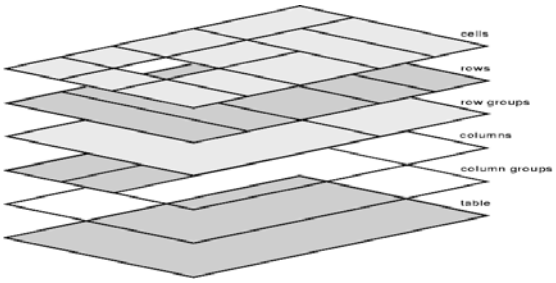
Ha a CSS:

```
table { empty-cells:hide; },
```

akkor a megjelenítés:



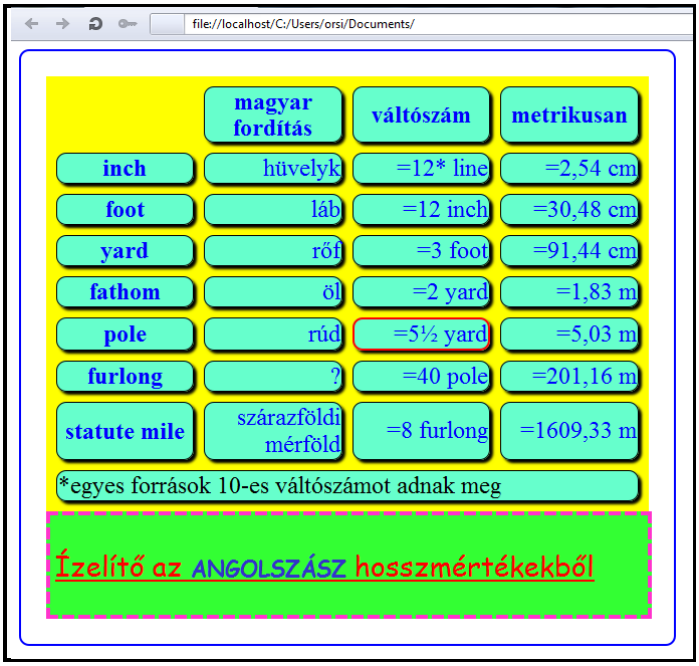
Megjegyzés: A belső táblázatok megjelenítési rétegei – a fedési sorrend szerint – az alábbiak szerint épülnek fel:



Ha a táblázatnak, az oszlopoknak, soroknak, és celláknak is van saját hátterük, akkor a fedési sorrend alapján egyidejűleg több is látszódhat közülük a *border-collapse:collapse* és/ vagy az *empty-cells:hide* esetén.

Például a legutolsó kódoláshoz adjunk a celláknak is saját hátteret:

```
th, td { background-color:#6FC; }
```



A felhasznált táblázat-specifikus tulajdonságok:

empty-cells	üres cellák megjelenítése
table-layout	táblázat alaprajzának kialakítása
caption-side	táblázatcím helyzete
border-collapse	cellaszegélyek összevonása
border-spacing	cellaszegélyek térköze

3.22. Túlnyúlás

Az **overflow** (túlnyúlás, túlcsordulás) tulajdonság azt specifikálja, hogy milyen legyen a megjelenítés, ha egy tartalom túlnyúlik a befoglaló doboznak a tartalom számára rendelkezésre álló (*content-box*) területén.

Figyelem! Az *overflow* tulajdonság blokkszintű (*block*) és soron belüli blokk (*inline-block*) elemeknél alkalmazható.

Az *overflow* tulajdonság lehetséges értékei: *visible* (alapértelmezett), *hidden, scroll, auto, no-display, no-content.*

- *visible:* a tartalom akár belefér a befoglaló doboz *content-box* részébe, akár azon kívülre is túlnyúlik, teljes mértékben megjelenik
- *hidden*: a tartalomnak csak a befoglaló doboz *content-box* részébe eső darabja jelenik meg, az azon kívül eső rész nem látszik
- *scroll*: a tartalomnak a befoglaló doboz *content-box* részébe eső darabja és a böngésző által alkalmazott görgető mechanizmus jelenik meg, mellyel a tartalom görgethető
- *auto:* ennek az értéknek a viselkedése böngészőfüggő, de valamilyen görgető mechanizmust hoz létre a túlcsorduló doboznak
- *no-display:* ha a tartalom nem illeszkedik a *content-box*-ba, az egész doboz el lesz távolítva, mintha *display:none* lenne specifikálva
- *no-content:* ha a tartalom nem illeszkedik a *content-box*-ba, az egész tartalom el lesz rejtve, mintha a *visibility:hidden* lenne specifikálva.

Megjegyzés: A jelenlegi böngészőverziók a *no-display* és *no-content* értékeket nem értelmezik, az *auto-ra* pedig a *scroll* görgetősávot alkalmazzák.

Az *overflow* tulajdonság egy összevont (*shorthand*) alak, mely az *overflow-x* vízszintes (*x*-tengely) irányú, és az *overflow-y* függőleges (*y*-tengely) irányú túlnyúlás kezelését definiálja. Ha csak egy érték van megadva, akkor az mind az *overflow-x*-re, mind az *overflow-y*–ra vonatkozik. Ha két érték van megadva, akkor az első az *overflow-x*-et, a második az *overflow-y*–t jelenti. Egy böngésző elvileg egyszerre több görgető mechanizmust is használhat, pl. egy fajtát a vízszintes túlfolyásra és egy másik fajtát a függőleges túlfolyásra.

Figyelem! Ha görgetősáv helyeződik egy elem széléhez, akkor a belső szegélyél (*border edge)* és a kitöltés külső éle *(padding edge)* közé illeszkedik be. A görgetősáv által elfoglalt hely befolyásolja a méretek kiszámítását, ugyanis takarni fogja a tartalom egy részét.

Példa a túlnyúlás kezelésének bemutatására:

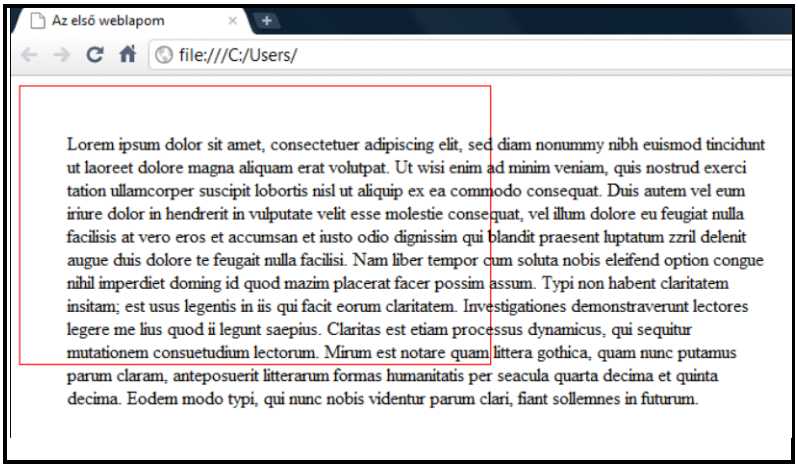
- legyen egy *400px* x *240px*-es *div* dobozunk, amit a jól láthatóság érdekében piros szegélylyel látunk el

- ebbe a dobozba egy *lipsum* kitöltőszöveget helyezünk el, persze úgy, hogy teljesen ne férjen el a *div*-ben, tehát legyen *600px* széles, és *40 px*-es margóval eltoljuk a kezdetét.

A HTML-kód fej részében pirossal szerepel a stílus:

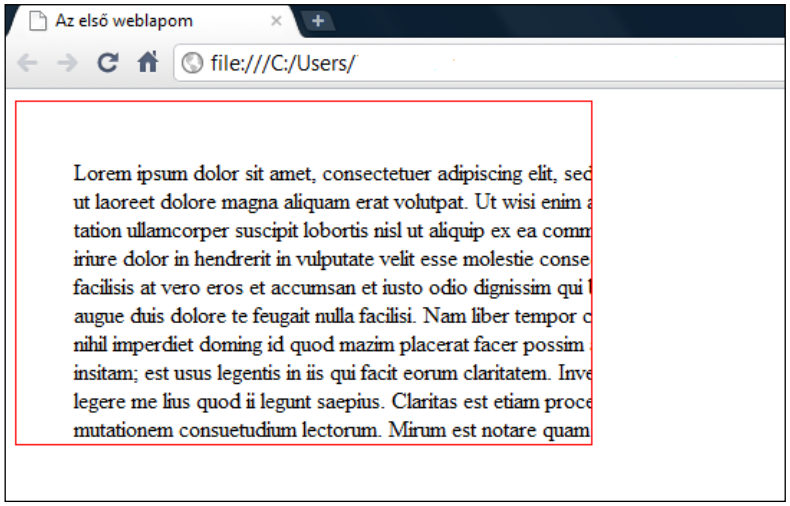
```
<!doctype html>
<html lang="hu">
  <head>
     <meta charset="utf-8">
     <title>Az első weblapom</title>
     <style>
     div { width:400px; height:240px; border:thin solid red; }
     p { width:600px; margin-top:40px; margin-left:40px; }
     </style>
  </head>
  <body>
     <div><p>Lorem ipsum dolor ……in futurum.</p></div>
  </body>
</html>
```

A böngészők az *overflow* tulajdonság nélkül ezt mutatják:



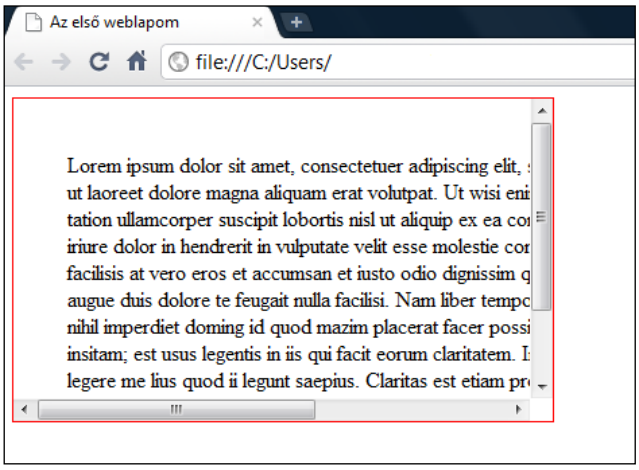
Ha az *overflow:visible* alapértelmezett értéken van, változatlan marad a megjelenítés. Ha a *hidden* értéket adjuk meg, csak a *div* dobozban lévő rész látszódik a szövegből:

```
div { width:400px; height:240px; border:thin solid red;
      overflow:hidden; }
```



Ha a *scroll* értéket adjuk meg, csak a *div* dobozban lévő rész látszódik görgetősávokkal ellátva (melyek takarnak a szövegből):

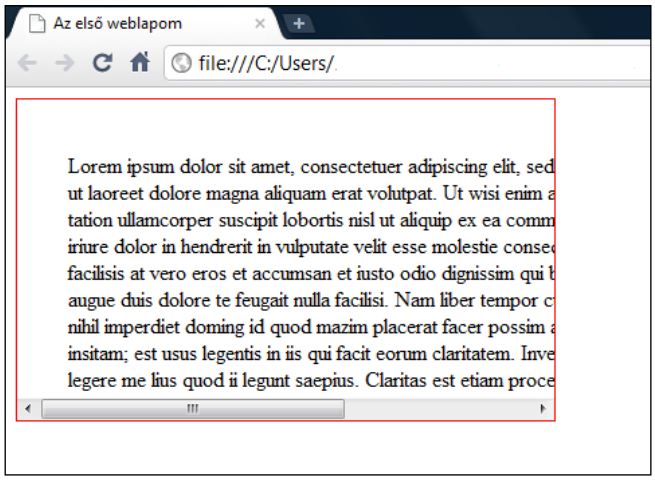
```
div { width:400px; height:240px; border:thin solid red;
      overflow:scroll; }
```



Ha az *auto* értéket adjuk meg, ugyanaz a megjelenítés, mint *scroll* esetén.

A vízszintes és függőleges túlnyúlást különválasztva és eltérő értékkel definiálva:

```
div { width:400px; height:240px; border:thin solid red;
      overflow-x:scroll; overflow-y:hidden; }
```



Láthatóan a függőleges görgetősáv eltűnik, és a szöveg csak vízszintesen mozgatható.

Kép esetében a túlnyúlás a szöveggel analóg módon kezelhető:

CSS:

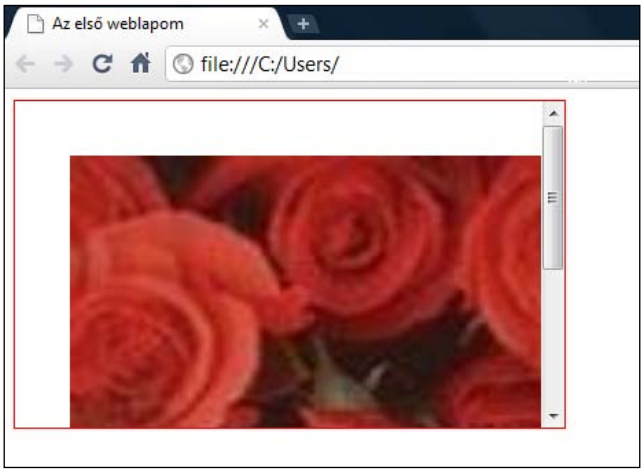
```
<style>
     div { width:400px; height:240px; border:thin solid red;
           overflow-y:scroll; overflow-x:hidden; }
     p { margin-top:40px; margin-left:40px; }
</style>
```

HTML:

```
<body>
    <div><p><img src="piros.jpg" width="400px" height="400px">
    </p></div>
</body>
```

Bekezdésbe raktuk a képet, hogy blokkszintű elemmé váljon (és persze legyen nagyobb a kép mérete a befoglaló *div*-énél).

A megjelenítés (a kép függőleges irányban mozgatható a dobozon belül):



Az **overflow-style** (túlnyúlás stílusa) tulajdonsággal specifikálható egy vagy több preferált görgetőmechanizmust. Szabvány szerinti lehetséges értékei: *auto* (alapértelmezett), *scrollbar*, *panner*, *move*, *marquee*. Az *overflow-style* értékeit preferencia-sorrendben, listaként felsorolva lehet megadni, a böngésző a listából az elsőt fogja alkalmazni, amelyet támogat. Ha egyiket sem támogatja, úgy viselkedik, mintha *auto* lenne az érték.

- *auto*: nincsen preferencia (a böngésző alapértelmezettje legyen – ez jelenleg a görgetősáv = scrollbar)
- *scrollbar*: keskeny csík, melyet az elem egyik vagy mindkét oldalához illesztenek be, és gyakran rákattintható nyílakkal és egy vonszoló csúszkával lehet az elem tartalmát fel-le vagy jobbra-balra vonszolni
- *panner*: az elem egyik sarkában látszódó négyszög, benne egy kisebb négyszöggel. A nagyobb négyszög képviseli az elem teljes tartalmát, a kisebb annak a látható részét. A kisebbik négyszöget mozgathatja a felhasználó, hogy ennek megfelelően mozgassa az elem tartalmát
- *move*: a felhasználó közvetlenül tudja mozgatni a tartalmat (nem közvetetten, mint a *scrollbar*-al vagy a *panner*-el). Tipikusan az egérkurzor kézzé vagy négy-nyilas keresztté változik, jelezve, hogy a felhasználó az egérrel vonszolhatja a tartalmat
- *marquee:* a tartalom autonóm módon, a felhasználó közreműködése vagy kontrollja nélkül mozog. A felhasználó, ha elég sokat vár, az egész túlnyúló tartalmat látni fogja. A mozgás iránya, jellege, sebessége, az ismétlődések száma az alábbiak szerint kódolható:
- *marquee-style* (a mozgatás jellege):
    - *scroll* (egyik oldalról indul és eltűnik a másik oldalon)
    - *slide* (a jobb oldalról elindul, a bal oldalon megáll, majd visszaugrik a jobb oldalra)

- *alternate* (oda-vissza mozog)

(Egy elem teljes tartalma egy darabban mozog, ha tehát pl. egy elem két szövegsorból áll, mindkét sor blokkszerűen együttesen halad.)

- *marquee-play-count* (a mozgatás száma):
  - tetszőleges pozitív egész szám
  - *infinite* (végtelen)

(Ha a mozgatás száma nagyobb mint 16, ill. végtelen, a böngésző 16-nál megállítja a mozgást – tehát csak 1 és 16 közötti számot érdemes megadni.)

- *marquee-direction* (mozgatás iránya):
  - *forward* (előre): úgy mozog a szöveg, hogy az eredetileg elrejtett rész a normál olvasási iránynak megfelelően jelenjen meg
  - *reverse* (visszafelé, fordított): a *forward* ellenkezője

- *marquee-speed* (mozgatás sebessége):
  - *slow* (lassú)
  - *normal* (normális)
  - *fast* (gyors)

(A *slow, normal* és *fast* értékek a böngészőtől és a mozgatott tartalom típusától függnek.)

Figyelem! Régebben az Internet Explorer HTML-címkeként vezette be a *marquee*-t, és ezt több böngésző is átvette tőle, azonban a W3C HTML szabványának semelyik verziójába soha nem került bele (viszont sok weblapon találkozni a használatával). A CSS3-ban szabványos megjelenítési tulajdonságnak szánták, kérdés azonban, hogy mikor kap széleskörű támogatást - vagy úgy jár, mint a szövegdíszítésnél a *blink* (jelenleg használhatatlan a szabványos kódolás). Alternatív megoldást lásd az *Animáció* fejezet végén.

3.23. Egyes elemekhez kapcsolódó kurzor-megjelenítés

A különböző elemekhez a **cursor** tulajdonsággal rájuk jellemző vagy róluk információt közvetítő kurzor-alakzatok kódolhatók, melyek általában a kurzort az elem fölé mozgatva, a *:hover* dinamikus ál-osztály kijelöléssel jelentkeznek.

A *cursor* tulajdonság lehetséges értékei:

- *auto:* a böngésző határozza meg a kurzor alakját (ferde nyíl)
- *default:* az operációs rendszer határozza meg a kurzor alakját (ferde nyíl)
- *help:* segítség áll rendelkezésre (nyíl mellett kis kérdőjel)
- *pointer:* link-re utal (kézfej kinyújtott mutatóujjal)
- *wait:* a program dolgozik, a felhasználónak várnia kell (homokóra, forgó kör)
- *progress:* a program dolgozik, de a felhasználó közbeavatkozhat (homokóra, forgó kör)
- *text:* választható szöveg (függőleges vagy ferde vékony vonal, mint a szövegbeírásnál)
- *move:* mozgatható elem (kereszt mind a négy végén nyíllal)
- *not-allowed:* a kívánt művelet nem hajtható végre (piros szegélyű fehér kör pirossal ferdén áthúzva
- *no-drop:* a vonszolt elem nem ejthető le (piros szegélyű fehér kör pirossal ferdén áthúzva)
- *crosshair:* raszterképes kijelölési mód (vékony kereszt)

- *all-scroll:* bármilyen irányban görgethető elem (négy irányú nyíl)

- *copy:* másolható elem (nyíl mellett kis kereszt – Safari-nál változatlan marad a cursor)

- *cell:* kiválasztható cella vagy cellacsoport (vastag fehér kereszt - Safari-nál változatlan marad a cursor)

- *alias:* létrejövő elem (nyíl mellett egy kis görbe nyíl - Safari forgó kört mutat)

- *none:* eltűnik a kurzor (Safari és Opera nem veszi figyelembe)

- *zoom-in, zoom-out:* nagyítás, kicsinyítés (nagyító + vagy – jellel a közepén; IE és Opera nem, Firefox – *moz-*, Chrome és Safari –*webkit-* előtaggal értelmezik)

- *context-menu:* menü érhető el az elemhez (nyíl mellett vagy helyett kis menü-ábra - Firefox, Chrom és Safari nem értelmezik)

- *col-resize:* oszlop vízszintes átméretezése (2 vízszintes nyíl között 2 függőleges vonal)

- *row-resize:* sor függőleges átméretezése (2 függőleges nyíl között 2 vízszintes vonal)

Az oldalirányú átméretezések jelölése az égtájak kezdőbetűivel történik, azaz:

e = east = keletre = jobbra

w = west = nyugatra = balra

n = north = északra = fel

s = south = délre = le

se = south-east = dél-keletre = jobbra le

nw = north-west = észak-nyugatra = balra fel

sw = south-west = dél-nyugatra = balra le

ne = north-east = észak-keletre = jobbra fel

Fentiek alapján egy adott irányban az átméretezhetőség jelölésének értékei: *e-resize, w-resize, n-resize, s-resize, se-resize, nw-resize, sw-resize, ne-resize* (vízszintes, függőleges vagy ferde fehér vonal mindkét végén nyílheggyel).

Ha kétirányú az átméretezhetőség, akkor az értékek: *ew-resize, ns-resize, nesw-resize, nwse-resize* (vízzintes, függőleges vagy ferde fehér vonal mindkét végén nyílheggyel).

A kurzor-alakzatok egy HTML-es <*div*> alábbi CSS-kódolásával megtekinthetők:

```
div { width:.........px; height:.........px; background-color.............; }
div:hover { cursor:.........; }
```

<u>A kurzor megjelenítéséhez felhasznált CSS tulajdonság:</u>

cursor	a kurzor megjelenítése

3.24. Egérkurzorral egyes elemek megjelenítésének a megváltoztatása

A pszeudo-osztály kijelölőknél említett *:hover* –hez (rámutatás egérrel) és *:active* –hoz (aktiválás lenyomva tartott bal egérgombbal) szinte mindenféle formázási lehetőség rendelhető. Amikor az egérrel a kijelölt elem fölé állunk, ill. lenyomva tartjuk az egérgombot (nem

kell kattintani!) az elem egy előre megadott új formázási állapotba kerül. A *:hover* és *:active* önmagában ugrásszerű változást eredményez, a folyamatos átmenetet a *transition* tulajdonsággal (lásd következő alfejezetet), a többszöri átmenetet az animációval (lásd a következő utáni alfejezetet) kiegészítve lehet létrehozni.

3.24.1. Hirtelen átmenet

A négy címsoros szövegünket a lekerekített sarkokkal foglalkozó fejezetben egy amorf idomba formáztuk, ezt változtatjuk most meg egérrel rámutatással.

HTML:

```
<div id="amorf">
      <h1>Helló világ!</h1>
      <h2>Legyen szép napod!</h2>
      <h3>Még hozzáírtunk valamit</h3>
      <h4>És még valamit</h4>
</div>
```

CSS:

Az eredeti formázás:

```
#amorf {
      width:300px;
      background-color:yellow;
      color:red;
      padding:30px 0px 30px 100px;
      border-radius:2em 14em 4em 8em / 3em 12em 6em 10em; }
```

(A *padding* és *border-radius* tulajdonságok értékeit összevontan adtuk meg.)

Az új formázás:

```
#amorf:hover {
      width:200px;
      background-color:red;
      color:teal;
      padding:10px 50px 40px 60px;
      border-radius:3em 12em 8em 12em / 5em 10em 8em 12em;
      margin:200px; }
```

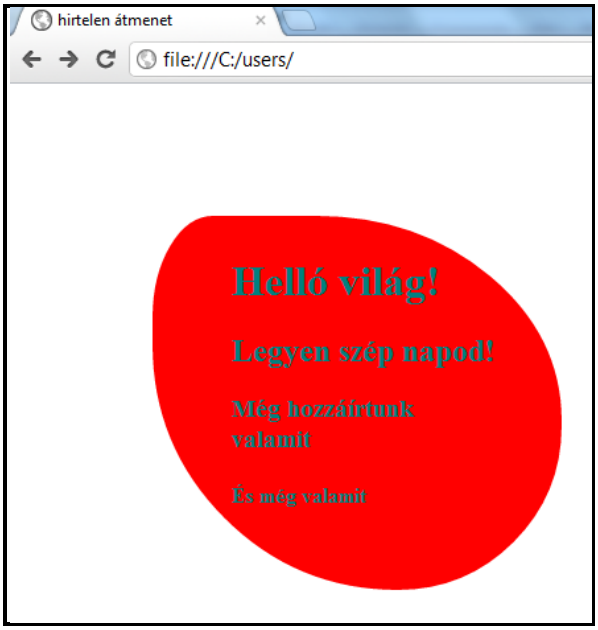
Az alakzat szélességét, háttérszínét, betűszínét, a szöveg alakzatban elfoglalt helyzetét, a kontúr lekerekítési sugarait és az alakzat pozícióját egyaránt módosítottuk.

Megjegyzés: Az *:active* teljes mértékben a fentiekben leírt módon kódolható, és ugyanezt a hatást váltja ki lenyomva tartott egérgomb esetén.

Ha csak az *#amorf:hover* kijelölést használnánk, akkor - ha az egérkurzor a szöveg fölé kerül - a formázatlan HTML szöveg ugrana az *#amorf:hover*-hez rendelt megjelenítésbe.

Ha az eredeti *#amorf*-os formázás és az *#amorf:hover*-es egyaránt szerepel a CSS-kódolásban, akkor - ha az egérkurzor a tartalom fölé kerül - az *#amorf*-os formázás átugrik a *#amorf:hover*-hez rendelt megjelenítésbe (a szöveg sortörése is egy mondatnál magától megváltozott, mert az új vízszintes méretben egy sorban nem fér el a szöveg).

Az egérmutatónak az elem fölé helyezésével (vagy *:active* kódolása esetén az egérgomb lenyomva tartásával) ilyenné válik tehát a négy címsoros szöveg:



Ha a *visibility:hidden;* tulajdonság/érték párt rendeljük az *#amorf:hover* kijelöléshez, akkor az egész elem ilyen módon időlegesen el is tüntethető. Nyilvánvalóan fordítva is működik a dolog, tehát elemek a láthatatlanságból előhívhatók, vagy nem látható helyről a pozíciójuk módosításával beúsztathatók, stb., és megjelenítésük számtalan egyéb tulajdonságának az értéke is módosítható.

A kurzorral létrehozott változtatás gyakori felhasználási területe pl. a **képváltás** (*roll-over image*), amikor a kurzorral egy kép fölé állva ugyanazon a helyen egy másik képre vált a weboldal. Kódolása:

HTML-kód:

```
<div>
    <img id="egy" src="…….jpg" width="……px" height="……px"
         alt="……">
    <img id="ketto" src="……..jpg" width="……px" height="……px"
         alt="……">
</div>
```

CSS-kód:

```
div img { position:absolute; }
#ketto:hover { opacity:0; }
```

A két kép egymásra van helyezve, a felső kép a kurzor hatására teljesen átlátszóvá válik – azaz az alsó kép válik láthatóvá. A képek *title* jellemzőit elhagytuk, hogy ne jelenjen meg a szöveges cím a kurzor hatására.

Megjegyzés: Az Internet Explorer 8 kedvéért a ***filter:alpha(opacity=0);*** - egyébként nem szabványos - kóddal ki lehet egészíteni az átlátszóságot (itt 0-100 közötti az értéktartomány): *#ketto:hover { opacity:0; filter:alpha(opacity=0); }*

3.24.2. Folyamatos átmenet

A *:hover*-el vagy *:active*-el létrehozott hirtelen ugrásszerű változások a **transition** tulajdonsággal folyamatos átmenetekké alakíthatók. Az átmenetek *(transitions)* olyan prezentációs effektek, melyek alkalmazása során egyes tulajdonságok az idő függvényében folyamatosan átmennek egy állapotból (értékből) egy másik állapotba (értékbe). Az átmenetek definiálásához két kötelezően megadandó és két opcionális paraméter áll rendelkezésre.

Kötelezően megadandó a **transition-property** (a változáson áteső tulajdonság vagy tulajdonságok listája) és a **transition-duration** (az átmenet vagy átmenetek időtartama).

Egyszerre több tulajdonság is változhat, a változtatandó tulajdonságokat a **transition-property** értékeiként, szóközzel, vesszővel felsorolva kell megadni (a sorrend közömbös). Ha pl. az előző alfejezetben létrehozott hirtelen átmenetnél előidézett változtatásokat akarjuk folyamatos változásként kódolni, akkor:

```
transition-property:width, background-color, color, padding,
                    border-radius, margin;
```

Az egyes változások időtartamát tulajdonságonként, szóközzel, vesszővel felsorolva, másodpercben kell megadni. A sorrend kötelezően a *transition-property* értékeinek sorrendjét követi, így tartozik minden változáshoz egyértelműen a változásának az időtartama. Különböző tulajdonságok változásaihoz különböző időtartamok tartozhatnak. A fenti tulajdonságokhoz találomra időtartamokat rendelve:

```
transition-duration: 4s, 10s, 5s, 8s, 8s, 3s;
```

A két paramétert beírva az *#amorf* tulajdonságai közé, folyamatos átmenettel változik meg a *:hover* hatására a négy mondatos amorf alakzat. Az átmenet végi állapot természetesen megegyezik a hirtelen átmenetnél kapottal. A *:hover*-hatás megszűntével az alakzat a folyamatos átmenet tükörképeként visszaalakul a kiinduló állapotba.

```
<style>
  #amorf {
      width:300px;
      background-color:yellow;
      color:red;
      padding:30px 0px 30px 100px;
      border-radius:2em 14em 4em 8em / 3em 12em 6em 10em;
      transition-property:width, background-color, color,
                          padding, border-radius, margin;
      transition-duration:4s, 10s, 5s, 8s, 8s, 3s;
  #amorf:hover {
      width:200px;
      background-color:red;
      color:teal;
      padding:10px 50px 40px 60px;
      border-radius:3em 12em 8em 12em / 5em 10em 8em 12em;
      margin:100px; }
</style>
```

A folyamatos átmenethez opcionálisan megadható két további paraméter, a **transition-delay** (átmenet késleltetése) és a **transition-timing-function** (az átmenet időzítésének függvénye).

A transition-delay (átmenet késleltetése) a tényleges átmenet kezdetének a tulajdonság megváltoztatására vonatkozó utasításhoz képesti időbeni eltolódását definiálja; 0 értéknél azonnal megkezdődik a változás (ez az alapértelmezett érték), egyébként a megadott pozitív értékű késleltetéssel indul el.

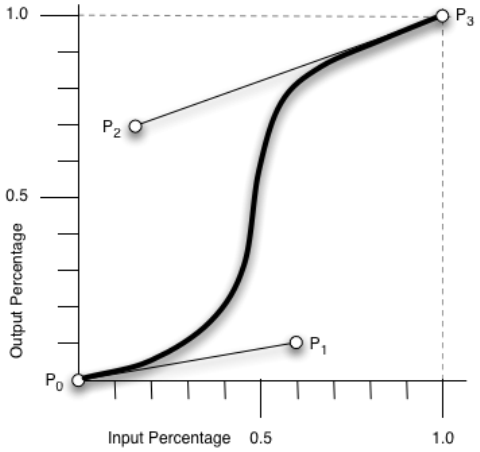
Ha negatív késleltetést definiálunk, természetesen nem indul el előbb a változás mint az azt kiváltó utasítás (hanem 0 késleltetéssel indul), de a tulajdonság kezdeti értéke nem az egyébként induló érték, hanem egy olyan közbenső állapot lesz, mintha a negatív késleltetés ideje alatt már bekövetkezett változásról indulna a folyamat.

Az egyes változtatott tulajdonságokra külön-külön előírható késleltetési érték, melyet szóközzel, vesszővel felsorolva, másodpercben kell megadni. A sorrend kötelezően a *transition-property* értékeinek sorrendjét követi, így tartozik egyértelműen minden változás kezdetéhez az esetleges késleltetése. Találomra késleltetve az átmenetünk tulajdonságait:

```
transition-delay: 5s, 4s, 8s, 10s, 7s, 3s;
```

A **transition-timing-function** (az átmenet időzítésének függvénye) arra vonatkozik, hogy az átmenet kezdeti és végpontja közötti közbenső időben milyen legyen a változó érték sebességváltozásának a módja.

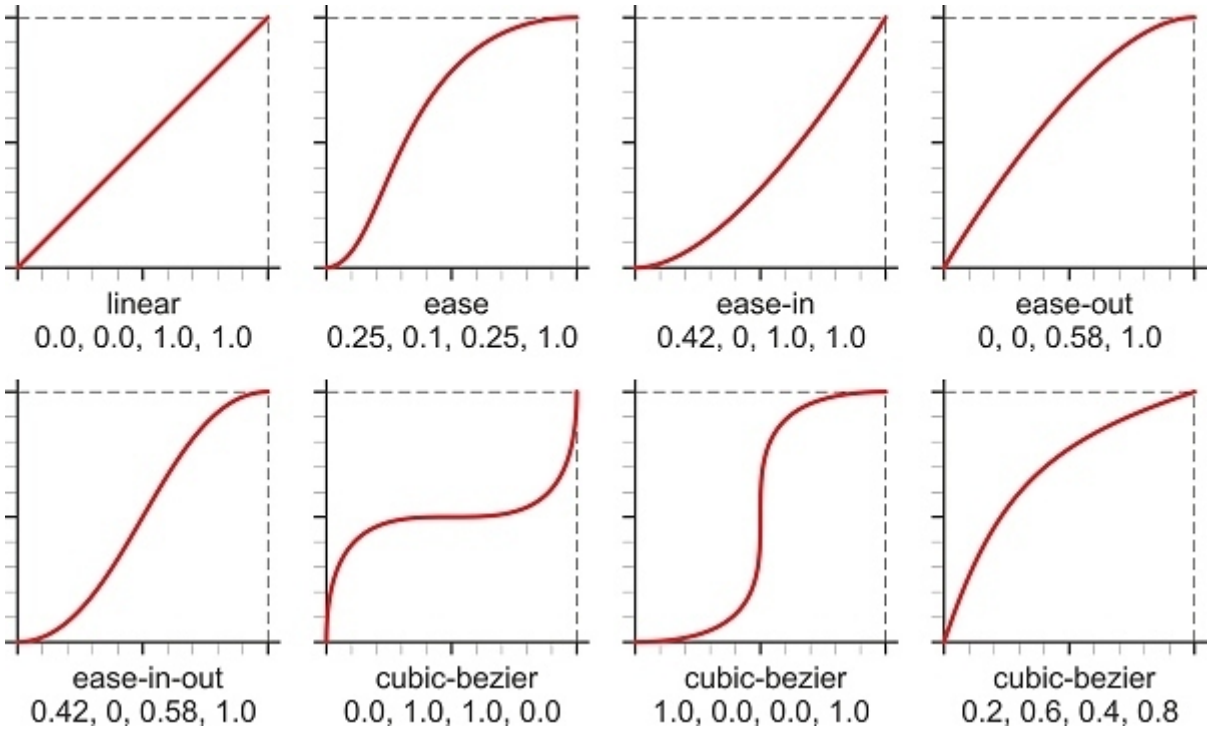
A folyamatos átmenet lefolyása négyzetes Bezier-görbével írható le, mely négy jellegzetes pontjával (P0, P1, P2, P3) definiálható. Ezek közül a P0 mindig (0,0) és a P3 mindig (1,1)**,** tehát csak a P1 és P2 pontok értékeit kell megadni, mely történhet névvel (*keyword*) vagy a görbe két pontja *x* és *y* koordinátájával (ezek csak 0 és 1 közötti értékek lehetnek):



Az *x*-tengelyen (*input* = független változó) az átmenet teljes idejéből (százalékosan) már eltelt idő van megadva, az *y*-tengely (*output* = függő változó) a változó tulajdonságnak a végső állapothoz képest elért (százalékos) értékét ábrázolja.

A változás a kezdő állapotból a *transition-timing-function* értékétől függetlenül ugyanannyi idő alatt ér a végső állapotba (azt csak a *transition-duration* szabja meg) – az eltérés a közbenső értékeknél jelentkezik, melyek eltérő ütemben változnak.

Az öt névvel megadható, ill. néhány általános alakú négyzetes Bezier-függvény görbéje és P1, P2 koordináta-értékeik (*x1, y1, x2, y2* sorrendben):



Névvel megadhatók tehát az alábbi átmenetek:

linear: ekvivalense *cubic-bezier(0.0, 0.0, 1.0, 1.0)* – egyenletes sebességű változás
ease: ekvivalense *cubic-bezier(0.25, 0.1, 0.25, 1.0)* – gyors kezdésből lassul a végére
ease-in: ekvivalense *cubic-bezier(0.42, 0, 1.0, 1.0)* – egyre gyorsul az elejétől kezdve
ease-out: ekvivalense *cubic-bezier(0, 0, 0.58, 1.0)* – hasonló az *ease*-hez, de kevésbé gyors az elején
ease-in-out: ekvivalense *cubic-bezier(0.42, 0, 0.58, 1.0)* – gyorsul az elején, lassul a végén

Az egyéb jellegű folyamatos változások *cubic-bezier(……)* alakban definiálhatók.

Megjegyzés: rövid átmeneteknél gyakorlatilag mindegy, melyik *transition-timing-function* értéket használjuk – kézenfekvő az alapértelmezett *ease* vagy a *linear* használata. Ha elfogadjuk az *ease*-t, egyáltalán nem kell ilyen esetben az átmenet jellegét kódolni.

Az egyes változtatott tulajdonságokra külön-külön előírható változási függvény, melyet szóközzel, vesszővel felsorolva kell megadni. A sorrend kötelezően a *transition-property* értékeinek sorrendjét követi, így tartozik egyértelműen minden változáshoz a változás jellege.

Találomra kiválasztva az átmenetünk tulajdonságainak jellegét:

```
transition-timing-function:cubic-bezier(0.2, 0.6, 0.4, 0.8),
     linear, ease, cubic-bezier(1.0, 0.0, 0.0, 1.0), linear,
     ease-in-out;
```

A **transition** tulajdonság a fenti négy (de legalább a kötelező kettő) tulajdonság összevont formában történő megadására szolgál, melyben szóközzel a kötelező sorrend: *transition-property transition-duration transition-timing-function transition-delay.* Ha több tulajdonság változásait adjuk meg egyszerre összevont alakban, vesszővel elválasztott listaként szerepelnek az egyes *shorthand* átmenet-egységek.

Az eddig tárgyalt folyamatos átmenet kódjának teljes összevont alakja tehát:

```
<style>
  #amorf {
     width:300px;
     background-color:yellow;
     color:red;
     padding:30px 0px 30px 100px;
     border-radius:2em 14em 4em 8em / 3em 12em 6em 10em;
     transition:width 4s cubic-bezier(0.2, 0.6, 0.4, 0.8) 5s,
              background-color 10s linear 4s,color 5s ease
              8s, padding 8s cubic-bezier(1.0, 0.0, 0.0, 1.0)
              10s, border-radius 8s linear 7s, margin 3s
              ease-in-out 3s; }
  #amorf:hover {
     width:200px;
     background-color:red;
     color:teal;
     padding:10px 50px 40px 60px;
     border-radius:3em 12em 8em 12em / 5em 10em 8em 12em;
     margin:100px; }
</style>
```

A *:hover*-hez hasonlóan történik az *:active*-el létrehozandó folyamatos átmenet.

Megjegyzések: Az Internet Explorer 8 és 9 nem értelmezik a *transition* tulajdonságot.

Az előző alfejezetben ismertetett hirtelen képváltás elegánsabb módja a folyamatos átmenettel történő váltás (*cross-fading images*). A két kép HTML-kódja változatlanul:

```
<div>
     <img id="egy" src="……….jpg" width="……px" height="……px"
          alt="……">
     <img id="ketto" src="……jpg" width="……px" height="……px"
          alt="……">
</div>
```

A CSS-kódolás:

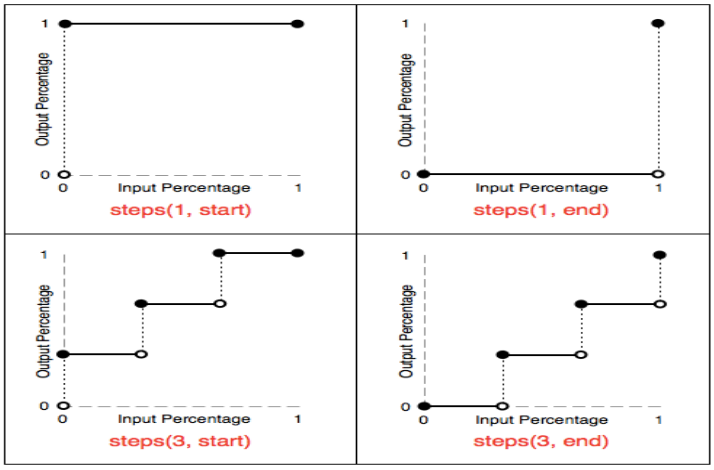
```
div img { position:absolute;
          transition:opacity 2s linear; }
#ketto:hover { opacity:0; }
```

A kurzor hatására két másodperc alatt olvad bele a felső kép az alsóba, ill. vált ismét vissza a kurzor elmozdítása után. A képek *title* jellemzőjét elhagytuk, hogy ezalatt ne jelenjen meg a kép címe.

A folyamatos átmenet speciális esete, és bizonyos szempontból közbenső helyzetet foglal el a hirtelen és folyamatos átmenet között, amikor a *transition-timing-function* ugrás-szerű változásokat ír le. A lépcsős-függvény (*stepping function*) az átmenet időtartamát és az átmenet kezdeti és végső értékének különbségét egyenlő szakaszokra osztja, így két paraméterrel definiálható: hány szakasz legyen (pozitív egész szám), és a szakaszok elején vagy végén történjen (*start* vagy *end*) az érték változása.

Amennyiben a *start* vagy *end* paraméter nincsen megadva, alapértelmezetten *end*-nek, tehát a szakaszok végén bekövetkező változtatásnak értelmezendő az lépcsős-függvény. Ha a szakaszok száma 1, az előző alfejezetben ismertetett hirtelen átmenetet valósítja meg.

A változás időbeli lefolyását egyszakaszos és háromszakaszos, ill. a szakasz(ok) elején vagy végén történő ugrás esetére ábrázolja az alábbi ábra:



Kódolás szempontjából nincsen különbség, hogy a négyzetes Bezier-függvény vagy a lépcsős-függvény szerinti átmenetet definiálunk – mindig a megfelelő átmenetet kell a *transition-timing-function* –ban megadni.

A hirtelen képváltás (*roll-over image*) megvalósítása *2 mp* késleltetéssel, a *transition* tulajdonság alkalmazásával tehát:

HTML:

```
<div>
    <img id="piros" src="piros.jpg" width="100px"
         height="100px" alt="piros rózsák">
    <img id="zold" src="zold.jpg" width="100px"
         height="100px" alt="zöld növények">
</div>
```

CSS:

```
div img { position:absolute;
          transition:opacity 2s steps(1, end); }
#zold:hover { opacity:0; }
```

Megjegyzés: A képek *title* jellemzőjét ezúttal elhagytuk, hogy ne jelenjen meg a kurzor kép fölé helyezésekor a szöveges cím.

3.24.3. Animáció

Az animáció az egyszeri folyamatos átmenet (*transition*) kiterjesztése. Míg a *transition*-nál egy vagy több tulajdonság megadott kezdeti és végső értéke(i) között a közbenső értékeket a böngésző határozza meg, és csak az időbeni lefolyás kódolható (a *transition-timing-function*, *duration*, *transition-delay* tulajdonságok értékeivel), az *animation* esetében a kezdő és végső érték(ek) közötti átmenetet finomabban/részletesebben lehet meghatározni, és az átmenetek többször (akár végtelen számban, azaz időkorlát nélkül, folyamatosan) megismételhetők.

Ennek megfelelően az animáció kódolása részben a folyamatos átmenetével analóg módon történik, részben pedig bővebb annál a részletesebb meghatározási lehetőségek miatt.

Táblázatos formában az egyszeri folyamatos átmenet és az animáció összehasonlítható tulajdonságai:

transition	animation
transition-property	animation-name
transition-duration	animation-duration
transition-timing-function	animation-timing-function
transition-delay	animation-delay
	animation-iteration-count
	animation-direction
	animation-fill-mode
	animation-play-state

A három analóg tulajdonság (*duration, timing-function* és *delay*) lehetséges értékei és azok értelmezése az átmenetnél és az animációnál megegyeznek.

Az *animation-iteration-count* (ismétlések száma) lehetséges értéke minden pozitív szám (nem csak egész szám lehet, tört szám esetén a ciklus tört részét hajtja végre az animáció) és a végtelen *(infinite)* - ennek az egyszeri átmenetnél nyilván nincsen analógja.

Az *animation-direction* (animáció iránya - az egyszeri átmenetnél szintén nincsen analógja) lehetséges értékei:

- *normal:* alapértelmezett, a kódolásnak megfelelően zajlik az animáció
- *reverse*: a kódolással ellentétes irányban történik az animáció – még az *animation-timing-function* is „megfordul”, pl. az eredeti *ease-in* megváltozik *ease-out* -ra
- *alternate*: a páratlan számú ciklusok *normal*, a páros számúak *reverse* módon játszódnak le
- *alternate-reverse:* a páratlan számú ciklusok *reverse*, a páros számúak *normal* módon játszódnak le

Az *animation-fill-mode* (kitöltési mód) tulajdonság határozza meg, hogy az animáció időtartamán kívül eső időben (vagyis előtte és utána) milyen értékeket alkalmazzon az animáció. Tekintettel arra, hogy a kezdeti és befejező értékek kötelezően adottak, az *animation-fill-mode* ezek felülírását teszi lehetővé. Lehetséges értékei:

- *none:* alapértelmezett, nincsen az animáció kezdő és befejező értékeire járulékos előírás
- *forwards:* az animáció *animation-iteration-count* által megszabott befejeződésekor kétféle értékeken állapodhat meg:

a) ha egész szám az ismétlések száma, akkor az utolsó ciklus befejező értékét veszi fel

b) ha nem egész szám az ismétlések száma, akkor nem a ciklus befejező, hanem a kezdeti értékeit veszi fel

- *backwards:* az *animation-delay*, tehát a késleltetés alatt az animáció az *animation-direction* tulajdonság *normal* vagy *alternate* értéke esetén a *from* kulcs képkocka, *reverse* vagy *alternate-reverse* értéknél a *to* kulcs képkocka értékeket alkalmazza
- *both:* mind a *forwards*, mind a *backwards* szabályokat követi, azaz mindkét irányban kiterjeszti az animációs tulajdonságokat

Megjegyzés: A *to* és *from* kulcs képkockák - és egyáltalán a kulcs képkockák - tárgyalására a következőkben kerül sor.

Az *animation-play-state* (lejátszási állapot) két lehetséges értéke a *running* (alapértelmezett - az animáció folyamatban van) és a *paused* (animáció megállítva). Az újra-

indítás is a *running* –al történik, ekkor az animáció a megállított értékektől, nem pedig az elejéről folytatódik.

Az animáció során a *transitions* átmenet igazi meghaladása a közbenső értékek definiálhatósága, mely egy *@keyframes* deklarációval kódolható. Ez közvetlenül nem szerepel a fenti táblázatban, viszont ezt kapcsolja az animációs tulajdonságokhoz a táblázatban szereplő *animation-name* (az animációt azonosító, általunk választott megnevezés).

A *@keyframes* (kulcs képkockák) deklaráció felépítése:

```
@keyframes animation-name {
from  { tulajdonság(ok):kezdeti értéke(i); }
……................................
……................................
to  { tulajdonság(ok):végső értéke(i); }
}
```

Egy teljes animációs ciklusnak a kezdő pontja a *from* – amivel ekvivalens megjelölés a *0%* - végpontja a *to* - amivel ekvivalens megjelölés a *100%*, közbenső pontokat a 0% és 100% közé lehet beiktatni. A kezdő- és végpont animált tulajdonságainak és azok értékeinek a megadása kötelező, a beiktatott közbenső pontok száma tetszőleges, használatuk opcionális.

Szokás szerint egy egyszerű *div*-en (ami bármilyen tartalmat képvisel) mutatjuk be az animáció működését. A HTML-kódban tehát csak egy * * szerepel, ennek megjelenítését animáljuk CSS-el.

a) A ciklus kezdetekor legyen a *div* helyzete: *margin-left:25px; margin-top:100px;*
A ciklus végére mozogjon jobbra és lefelé: *margin-left:270px; margin-top:180px;*
Ha nem adunk meg közbenső értéket, a böngésző számítja ki azokat az *animation-timing-function* alapján, de a példa kedvéért megadunk a ciklus 60%-ára vonatkozó helyzetet is: *margin-left:100px; margin-top:150px;* (ekkor a kezdőpont és a ciklus 60%-a, ill. a 60% és végpont közötti pályát számolja az *animation-timing-function* alapján a böngésző). Több közbenső pont beiktatásával a pálya alakja szükség esetén elképzelésünk szerint tovább alakítható. A példa kedvéért a többi tulajdonságra is specifikálunk közbenső (60%-os időponthoz tartozó) értékeket.

b) A ciklus kezdetekor legyen a *div* mérete: *width:20px; height:20px;*
A ciklus végére nőjön meg a tízszeresére: *width:200px; height:200px*;
Adjunk meg a 60%-hoz is egy értéket: *width:120px; height:120px;*

c) A ciklus kezdetekor legyen a háttér színe piros: *background-color:red*;
A ciklus végére legyen a háttérszín kék: *background-color:blue;*
A ciklus 60%-ánál pedig legyen zöld: *background-color:green;*

d) A ciklus elején legyen az alakja egy négyzet: *border-radius:0%;*
A ciklus végére menjen át kör alakba: *border-radius:50%;*
A ciklus 60%-ánál a sarkok lekerekítése legyen 20%: *border-radius:20%;*

e) Az animációnk neve (*animation name*) legyen: *pelda*

Fentiek alapján a *@keyframes* deklaráció kódolása:

```
@keyframes pelda {
    0% { background-color:red; border-radius:0%; width:20px;
         height:20px; margin-left:25px; margin-top:100px; }
   60% { background-color:green; border-radius:20%;
         width:120px; height:120px; margin-left:100px; margin-
         top:150px; }
 100% { background-color:blue; border-radius:50%; width:200px;
         height:200px; margin-left:270px; margin-top:180px; }
}
```

Figyelem! A cikluson belüli pontokat megadó % jelet kötelező - szóköz nélkül - kiírni. A kódolás végén a dupla kapcsos zárójel nem sajtóhiba – az egyik zárja az animált tulajdonságok ciklus végi értékeit, a másik pedig a *@keyframes* deklarációt.

Az animációs tulajdonságok kódolása az átmeneteknél már megismert módon történik, de kibővül a csak az animációra értelmezhető tulajdonságokkal (ciklus ismétlődési száma, az animáció iránya, lejátszási állapot, kitöltési mód):

```
div {
   animation-name:pelda;
   animation-timing-function:linear;
   animation-iteration-count:infinite;
   animation-duration:10s;
   animation-direction:alternate;
   animation-play-state:running;
   animation-delay:0s;
   animation-fill-mode:none; }
```

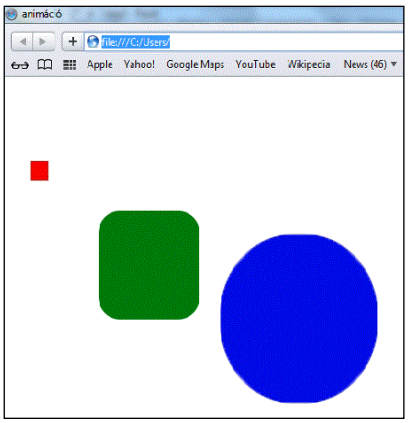
A fenti kódolás az animált tulajdonságok egyenletes sebességű, 10 másodperc ciklusidejű, időkorlát nélkül ismétlődő oda-vissza változását írja elő indulási késleltetés nélkül. A késleltetés, lejátszási állapot és kitöltési mód az alapértelmezett értékekkel van megadva, tehát a kódolásból ebben a specifikációban el is hagyhatóak.

Figyelem! A Firefox és az Internet Explorer a 10-től *prefix* nélkül, a Chrome, Opera és Safari pedig csak *–webkit-* előtaggal támogatja az animációt. Fentiek alapján még egy bizonyos ideig a szabványos mellett böngésző-specifikus előtaggal is kódolni kell:

```
<style>
  @-webkit-keyframes pelda {
    0% { background-color:red; border-radius:0%; width:20px;
         height:20px; margin-left:25px; margin-top:100px; }
   60% { background-color:green; border-radius:20%;
         width:120px; height:120px; margin-left:100px; margin-
         top:150px; }
 100% { background-color:blue; border-radius:50%; width:200px;
         height:200px; margin-left:270px; margin-top:180px; }
      }
  @keyframes pelda {
    0% { background-color:red; border-radius:0%; width:20px;
         height:20px; margin-left:25px; margin-top:100px; }
   60% { background-color:green; border-radius:20%;
         width:120px; height:120px; margin-left:100px; margin-
         top:150px; }
 100% { background-color:blue; border-radius:50%; width:200px;
         height:200px; margin-left:270px; margin-top:180px; }
      }
```

```
div {
      -webkit-animation-name:pelda;
      animation-name:pelda;
      -webkit-animation-timing-function:linear;
      animation-timing-function:linear;
      -webkit-animation-iteration-count:infinite;
      animation-iteration-count:infinite;
      -webkit-animation-duration:10s;
      animation-duration:10s;
      -webkit-animation-direction:alternate;
      animation-direction:alternate; }
</style>
```

A megjelenítés a ciklus elején, 60%-ánál és a végén:



Az animált tulajdonságok értékei összevontan is megadhatók, ekkor a tulajdonságok kötelező sorrendje: *animation-name*, *animation-duration*, *animation-timing-function*, *animation-delay*, *animation-iteration-count*, *animation-direction*, *animation-fill-mode*, *animation-play-state*, az értékek pedig egy szóközzel, vessző nélkül felsorolva adandóak meg.

A példánkban a szabványos összevont kódolás tehát:

```
div { -webkit-animation:pelda 10s linear 0s infinite alternate
                        none running;
      animation:pelda 10s linear 0s infinite alternate none
                        running; }
```

Figyelem! Az *animation* összevont alakjában jelenleg az *animation-play-state* egyik értékét sem értelmezik a Firefox és Internet Explorer böngészők, ezért használatuk során a teljes animáció meghiúsul. Így a rövid alak utolsó értékét egyelőre el kell hagyni a kódból – amennyiben az alapértelmezett *running* értéket specifikálnák, az elhagyás nem jelent gondot, ha a *paused* értékre lenne szükség, akkor a működő, részletezett alakot kell használni, hogy lefusson az animáció.

A *@keyframes* rész változatlan marad – nem kell feltétlenül a kódolásban közvetlenül az *animation* előtt vagy után elhelyezkednie, bár áttekinthetőbb a kód, ha egymást követik, és nem ékelődik közéjük más formázási feladat.

Megjegyzés: Az animáció a fentiekbeni kódolás esetén külön meghatározó esemény nélkül, a weblap betöltésekor (az esetleges kódolt késleltetésnek megfelelően) automatikusan bekövetkezik. Amennyiben ezt a felhasználó valamilyen műveletéhez (pl. *hover*, *active*, stb.) akarjuk kötni, az alábbi kiegészítéssel lehetséges:

a) a részletezett animációs kódban az *animation-play-state* tulajdonságot *paused* értékre kell beállítani:

```
div {
        -webkit-animation-name:pelda;
        animation-name:pelda;
        -webkit-animation-timing-function:linear;
        animation-timing-function:linear;
        -webkit-animation-iteration-count:infinite;
        animation-iteration-count:infinite;
        -webkit-animation-duration:10s;
        animation-duration:10s;
        -webkit-animation-direction:alternate;
        animation-direction:alternate;
        -webkit-animation-play-state:paused;
        animation-play-state:paused; }
```

b) a kurzor lenyomva tartásával az *animation-play-state* tulajdonságot *running* értékre módosítjuk:

```
:active div {  -webkit-animation-play-state:running;
               animation-play-state:running;}
```

Az egérgombot lenyomva elindul az animáció, és addig tart, amíg a gomb lenyomva marad. A gomb felengedése esetén az éppen aktuális állapotában az animáció megáll, ismételt lenyomásával ebből az állapotából a félbehagyott ciklus folytatásaként továbbindul.

Ha a kurzornak a kép feletti alakváltozásával jelölni kívánjuk, hogy hatására valami történni fog, akkor a *div* CSS-kódjába még beírható:

```
div { ..............................
      ..............................
      ..............................
      cursor:pointer; }
```

Megjegyzés: Elvileg a tulajdonságok széles köre animálható, a gyakorlatban a böngészők között van némi eltérés ezügyben. Egy lista az általában animálható tulajdonságokról:

background-color, background-position (ha nem kulcsszóval van megadva), background-size (ha nem kulcsszóval van megadva), border-color, border-radius, border-spacing, border-width, color, font-size, font-weight, letter-spacing, line-height, margin, opacity, padding, text-indent, text-shadow, box-shadow (inner-outer között nem), vertical-align, visibility, word-spacing, z-index, min/max-width, min/max-height – és az ezután tárgyalt linear/radial-gradient (linear-ból radial-ba és viszont nem lehet, color-stop-ok száma és helyzete nem változhat).

Blink és ***marquee*** kiváltása animációval***:***

a) A szövegdíszítésnél említett villogás létrehozható a kiutált *text-decoration:blink;* kód helyett animációval is, a villogó szövegrész átlátszóságát ugrásszerűen - a lépcsős függvény alkalmazásával - változtatva *0* és *1* érték között. Legyen a HTML-kód:

```
<p>Mégis megoldható CSS-el a <span class="blink">villogás
                              </span> mindegyik böngészőben.</p>
```

CSS:

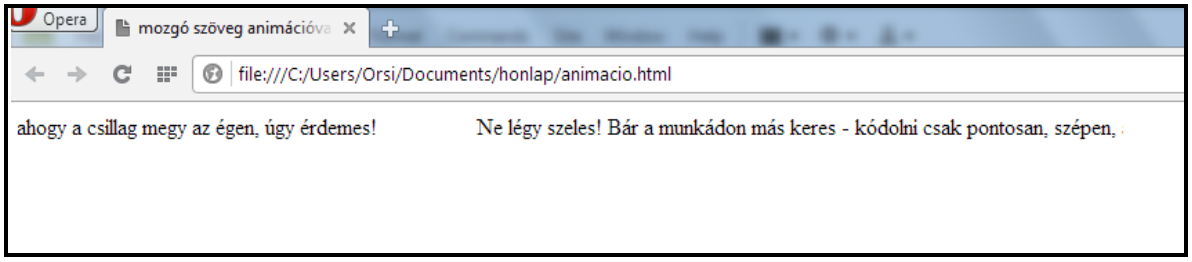
```
<style>
   p .blink {
      -webkit-animation:blink 1s steps(2, start) 0s infinite;
      animation:blink 1s steps(2, start) 0s infinite; }

   @-webkit-keyframes blink {
      0% { opacity:1; }
    100% { opacity:0; }
   }
   @keyframes blink {
      0% { opacity:1; }
    100% { opacity:0; }
   }
</style>
```

b) A túlnyúlásnál említett mozgó szöveg is létrehozható a működésképtelen *overflow-style:marquee;* helyett animációval. A *marquee-style:scroll; marquee-direction:forward; marquee-play-count: infinite;* (olvashatósági irányú mozgás, végtelen ismétlődés, egyik oldalról indulva a szöveg eltűnik a másik oldalon) kialakítás megvalósítása pl.:

```
<!doctype html>
<html lang="hu">
  <head>
    <meta charset="utf-8">
    <title>mozgó szöveg animációval</title>
    <style>
       div { width:800px; }
       .marquee {
             height: 30px;
             overflow: hidden;
             position: relative; }
       .marquee div {
             display: block;
             width: 200%;
             height: 30px;
             position: absolute;
             overflow: hidden;
          -webkit-animation: marquee 10s linear 0s infinite;
             animation: marquee 10s linear 0s infinite; }
             @-webkit-keyframes marquee {
                  0% { left: 0; }
                100% { left: -100%; }
               }
             @keyframes marquee {
                  0% { left: 0; }
                100% { left: -100%; }
               }
       .marquee span {
             float: left;
             width: 50%; }
      </style>
    </head>
```

```
  <body>
      <div class="marquee">
         <div>
            <span>Ne légy szeles! Bár a munkádon más keres
                  - kódolni csak pontosan, szépen, ahogy a
                  csillag megy az égen, úgy érdemes!</span>
            <span>Ne légy szeles! Bár a munkádon más keres
                  - kódolni csak pontosan, szépen, ahogy a
                  csillag megy az égen, úgy érdemes!</span>
         </div>
      </div>
  </body>
</html>
```

Egy pillanatfelvétel a folyamatosan futó szövegről:



3.25. Többhasábos elrendezés

Többhasábos elrendezés (*multi-column layout*, vagy röviden *multicol*) HTML-kódolással (rosszabb esetben) táblázatokkal, vagy (jobb esetben) egymás mellé helyezett *div* tárolóelemekkel hozható létre.

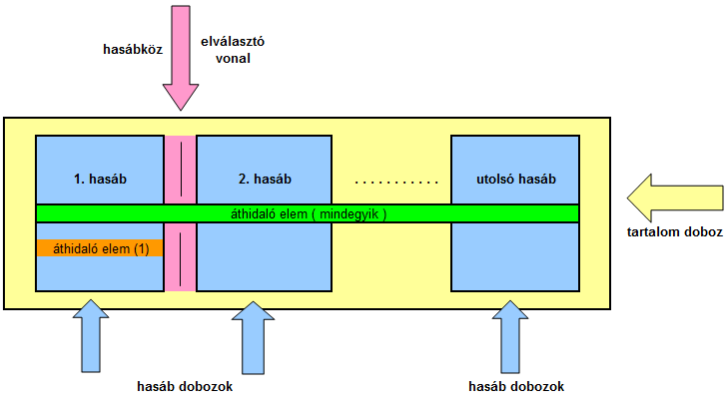
A CSS-el történő kialakítás fő előnye, hogy a tartalom rugalmasan áramlik az egyik hasábból a másikba, és a hasábok száma a megjelenítőeszközök méretének, ill. a tartalom bővülésének vagy csökkenésének a függvényében változhat.

Az elemi doboz-modellben a tartalom az adott elemhez tartozó tartalom-dobozban (*content box*) áramlik. A többhasábos elrendezés egy új típusú doboz, a hasáb-doboz (*column box*, vagy röviden *column*; column = oszlop, hasáb) fogalmát vezeti be, mely a tartalom és a tartalom-doboz között helyezkedik el, és a tartalom ezekben a hasáb-dobozokban áramlik.

A hasáb-dobozok sorban helyezkednek el, valamennyiük szélessége azonos, és az egy sorban lévők egyforma magasak. A szomszédos hasáb-dobozokat hasábköz (*column gap*) választja el egymástól, amely opcionálisan tartalmazhat elválasztó vonalat (*column rule*). Az egy sorban lévő valamennyi hasábköz és elválasztó vonal egyenlő.

A hasáb-dobozokra nem lehet definiálni tulajdonságokat, pl. saját hátteret, külső vagy belső margót, szegélyeket, stb., és csak akkor jelennek meg, ha van tartalmuk.

Áthidaló elem (*spanning element*) vagy egy hasábot, vagy az összeset fogja át, közbenső érték nem adható meg. A hasáboknak az áthidaló elem előtti tartalma kiegyenlítődik, utána tovább áramlik.



Figyelem! A *multicol* tulajdonságot a böngészők közül az Internet Explorer a 10-től, a Chrome, Firefox, Opera és Safari pedig csak saját *prefix*-es kódokkal értelmezik.

Az elrendezés két legalapvetőbb tulajdonsága a hasábok száma és a hasábszélesség:

column-count: a hasábok számát definiálja, értékei:

- *auto:* más tulajdonság szabja meg (pl. a hasábszélesség, ha annak nem *auto* az értéke)
- *pozitív egész szám:* a hasábok maximális számát adja meg, ha a hasábszélesség nem *auto*

column-width: hasábszélességet definiálja, értékei:

- *auto:* azt jelenti, hogy a szélességet más tulajdonság (pl. a hasábok száma, ha annak nem *auto* az értéke) határozza meg
- *szélesség:* pozitív érték; a tényleges hasábszélesség lehet szélesebb, hogy kitöltse a rendelkezésre álló teret, vagy keskenyebb, ha a rendelkezésre álló hely kisebb a megadott hasábszélességnél

Az esetek többségében csak a *column-width* vagy *column-count* egyike befolyásolja az elrendezést. Ha a *column-width* értéke nem *auto*, a *column-count* értéke nincs figyelembe véve. Az egyetlen eset, amikor mindkettő alakítja az elrendezést, olyan elemnél lehet, melynek a szélessége nincsen definiálva.

A *columns* egy rövid alak, mely a *column-width*-et és a *column-count*-ot (ebben a sorrendben) definiálja. Ha csak az egyikük van megadva, a kihagyott érték *auto*-t jelent.

Az elrendezés további, opcionális tulajdonságai a hasábköz és az elválasztó vonal:

column-gap (hasábköz): hossza egyenlő a hasábok hosszával (magasságával), szélessége helyet foglal el, tehát széttolja egymástól a szomszédos hasábok tartalmát. Lehetséges értékei:

- *normal:* ez az alapértelmezett, böngészőfüggő az értéke
- *szélesség:* pozitív szám (javasolt érték *1em*)

Ha pontos hasábszélességet akarunk definiálni, mind a *column-width*, mind a *column-gap* értékét meg kell adni.

column-rule (elválasztó vonal) a hasábköz közepén húzódik, hossza egyenlő a hasábok hosszával. A hátteret takarja, de helyet nem foglal el, tehát a többi rész elhelyezkedését nem változtatja meg. Ha szélesebb mint a hasábköz, be fog nyúlni a szomszédos hasábokba.

Az elválasztó vonal a (doboz)szegélyekhez hasonló módon formázható a következők szerint:

column-rule-width: bármilyen pozitív érték

column-rule-style: az elválasztó vonal stílusa a szegélystílusokkal analóg (lásd *border-style*) módon definiálható (de az *inset ridge*-nek, az *outset groove*-nak látszik), a *none* érték az elválasztó vonal szélességét 0-ra állítja

column-rule-color: az elválasztó vonal színe a(z előtér) színeknél megadott módokon definiálható (lásd *color*)

A **column-rule** egy rövid alak, mely a *column-rule-width*, *column-rule-style* és *column-rule-color* értékeket (ebben a sorrendben) tartalmazza – a kihagyott érték az alapértelmezettet jelenti.

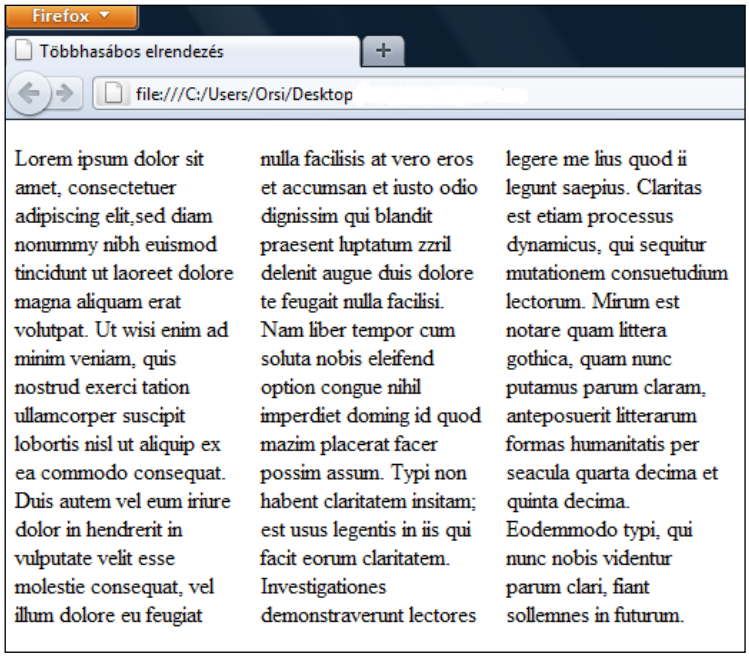
Példaként kezdjük el a *lipsum*-ot többhasábosan formázgatni. A HTML-kód ismert:

```
<p>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
diam nonummy nibh euismod tincidunt ut laoreet dolore magna
aliquam erat volutpat. Ut wisi enim ad minim veniam, quis
nostrud exerci tation ullamcorper suscipit lobortis nisl ut
aliquip ex ea commodo consequat. Duis autem vel eum iriure
dolor in hendrerit in vulputate velit esse molestie consequat,
vel illum dolore eu feugiat nulla facilisis at vero eros et
accumsan et iusto odio dignissim qui blandit praesent luptatum
zzril delenit augue duis dolore te feugait nulla facilisi. Nam
liber tempor cum soluta nobis eleifend option congue nihil
imperdiet doming id quod mazim placerat facer possim assum.
Typi non habent claritatem insitam; est usus legentis in iis
qui facit eorum claritatem. Investigationes demonstraverunt
lectores legere me lius quod ii legunt saepius. Claritas est
etiam processus dynamicus, qui sequitur mutationem
consuetudium lectorum. Mirum est notare quam littera gothica,
quam nunc putamus parum claram, anteposuerit litterarum formas
humanitatis per seacula quarta decima et quinta decima. Eodem
modo typi, qui nunc nobis videntur parum clari, fiant
sollemnes in futurum.
</p>
```

Először a szélességet (a vízszintes kiterjedést) határoljuk be, a magasság (függőleges kiterjedés) a tartalomnak megfelelően, szabadon alakul. Legyen a szélesség max. *500 px*, a hasábok száma *3* (ekkor a hasábszélességet a böngésző határozza meg):

```
<style>
      p {
            -moz-column-count:3;
            -webkit-column-count:3;
            column-count:3;
            max-width:500px; }
</style>
```

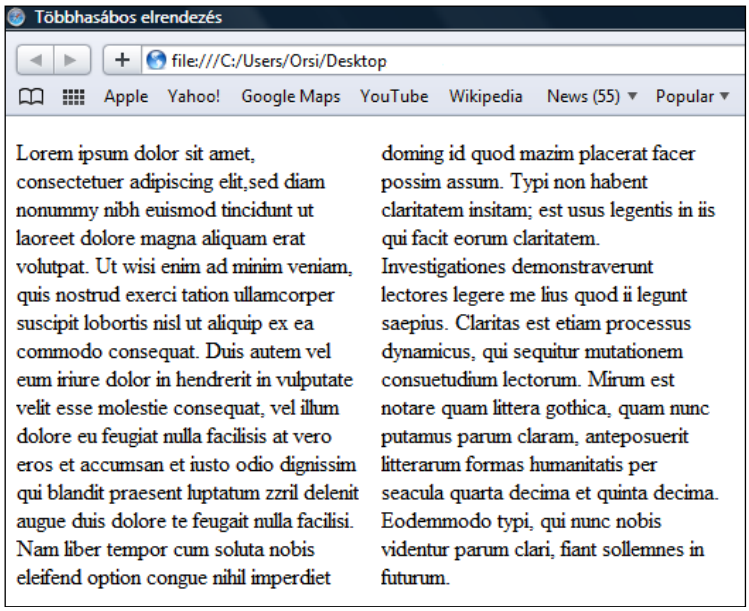
Ekkor a megjelenítés:



Most mi adjuk meg a hasábszélesség értékét is, ami legyen *200px*-t:

```
<style>
      p {
            -moz-column-count:3;
            -webkit-column-count:3;
            -moz-column-width:200px;
            -webkit-column-width:200px;
            column-count:3;
            column-width:200px;
            max-width:500px; }
</style>
```

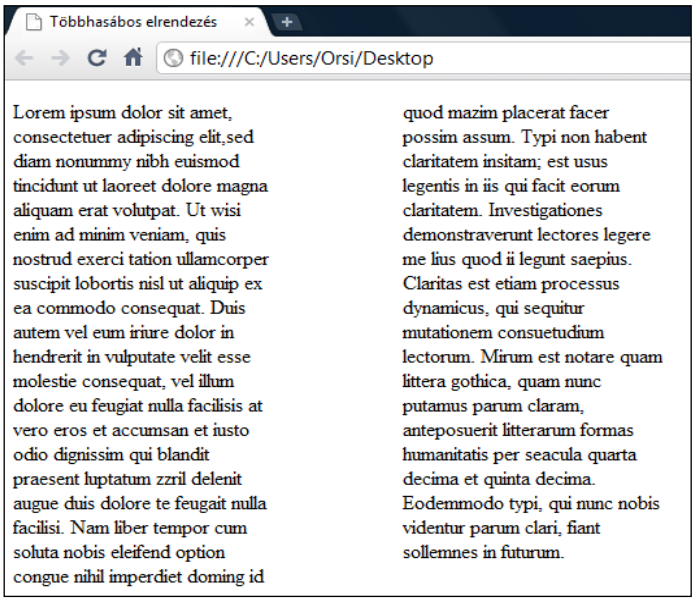
Ez a hasábszélesség érték felülírja a 3 hasábos, a böngésző által számított értéket, mivel csak 2 hasáb fér el az adott max. szélességben:



Definiáljunk *100px* hasábközt is (evvel még pont beleférünk a megadott max. szélességbe:

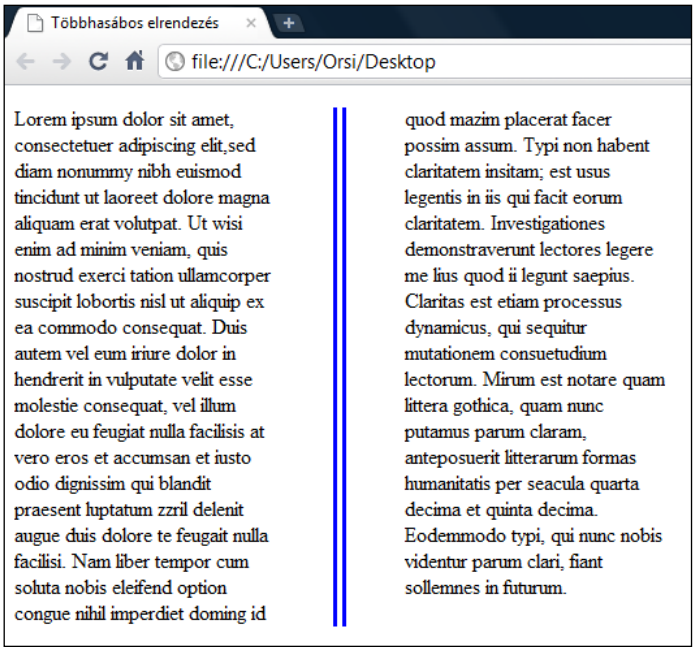
```
<style>
      p {
            -moz-column-count:3;
            -webkit-column-count:3;
            -moz-column-width:200px;
            -webkit-column-width:200px;
            -moz-column-gap:100px;
            -webkit-column-gap:100px;
            column-count:3;
            column-width:200px;
            column-gap:100px;
            max-width:500px; }
</style>
```

100px-es *column-gap* felett szétesik a többhasábos elrendezés és egyhasábossá válik, *100px* alatt szabadon beállítható a hasábköz és marad a kéthasábos megjelenítés:



Adjunk hozzá (kék, dupla, *10px* széles) elválasztó vonalat:

```
<style>
      p {
            -moz-column-count:3;
            -webkit-column-count:3;
            -moz-column-width:200px;
            -webkit-column-width:200px;
            -moz-column-gap:100px;
            -webkit-column-gap:100px;
            -moz-column-rule:10px double blue;
            -webkit-column-rule:10px double blue;
            column-count:3;
            column-width:200px;
            column-gap:100px;
            column-rule:10px double blue;
            max-width:500px; }
</style>
```



Megjegyzés: Az elválasztó vonal nem növeli meg a hasábköz szélességét.

Most korlátozzuk a többhasábos elrendezés max. magasságát pl. *300px*-re:

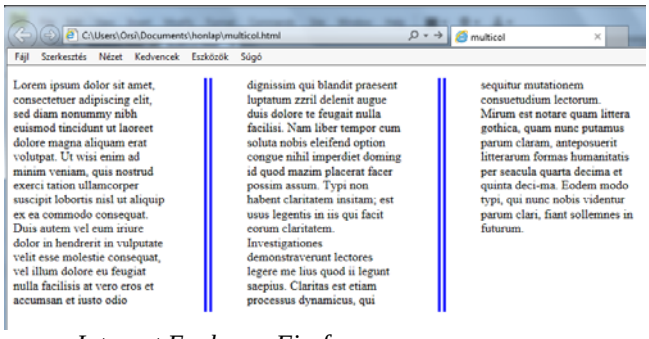
```
<style>
      p {
            -moz-column-count:3;
            -webkit-column-count:3;
            -moz-column-width:200px;
            -webkit-column-width:200px;
            -moz-column-gap:100px;
            -webkit-column-gap:100px;
            -moz-column-rule:10px double blue;
            -webkit-column-rule:10px double blue;
            column-count:3;
            column-width:200px;
            column-gap:100px;
            column-rule:10px double blue;
            max-width:500px;
            max-height:300px; }
</style>
```

A magasság korlátozása – azaz többszörös, ellentmondó kényszerek a hasábelrendezésre - már böngészőnként eltérő eredményt hoz:

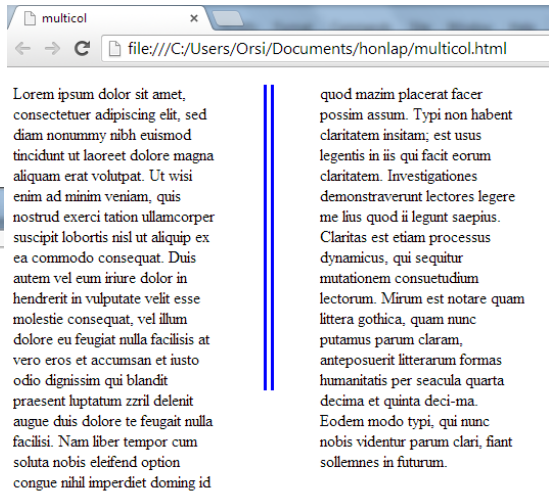
- az Internet Explorer-nél és Firefox-nál felülírta a korábbi hasábelrendezést, és a tartalom-doboz (a *p* doboz) méretének (*width* és *height*) megfelelően alakította ki a böngésző a hasábokat

- a *-webkit-* es böngészőknél (Chrome, Opera, Safari) viszont zavart okozott és hibás, kevert megjelenést eredményez

A kétféle megjelenítés:



Internet Explorer, Firefox



Chrome, Opera Safari

Az **áthidaló elem** (lásd a fejezet bevezető ábráját) vagy csak egy hasábot, vagy az összeset átíveli, melyet a *column-span* (hasáb átfogás) tulajdonság definiál. Lehetséges értékei:

- *none*: nem ível át több hasábot

- *all*: valamennyi hasábot átíveli

Egy címmel rendelkező többoszlopos szövegblokk formázását a HTML-dokumentumban a *lipsum* elé egy *h2* címsort beírva és a címet és szöveget egy *div*-be egybefogva az alábbiakban mutatjuk be:

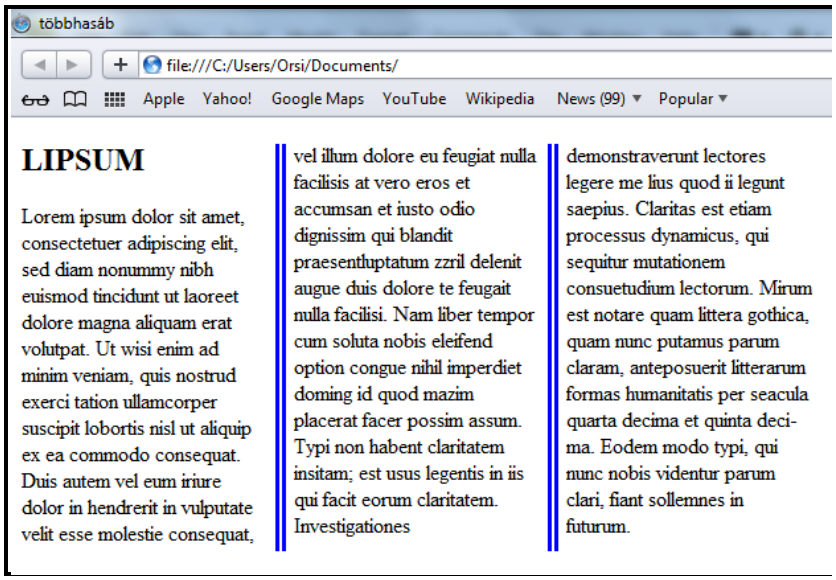
A HTML-kód:

```
<div>
      <h2>LIPSUM</h2>
      <p>Lorem ipsum dolor sit amet.......................in futurum.</p>
</div>
```

a) A CSS-kód, ha címsor nem ível át több hasábot:

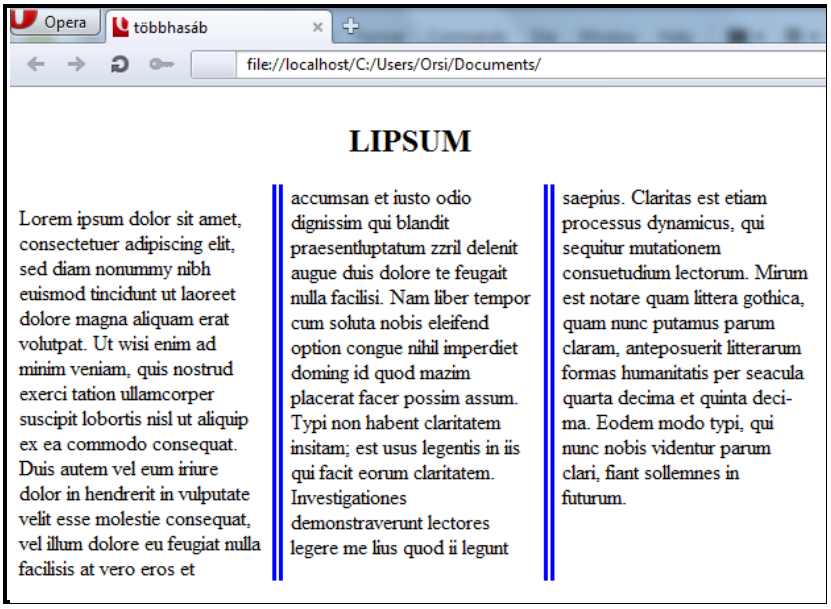
```
div {
      -webkit-column-count:3;
      -moz-column-count:3;
      -moz-column-gap:20px;
      -webkit-column-gap:20px;
      -moz-column-rule:8px double blue;
      -webkit-column-rule:8px double blue;
      column-count:3;
      column-gap:20px;
      column-rule:8px double blue;
      width:600px; }
h2 {
      -webkit-column-span:none;
      -moz-column-span:none;
      column-span:none;
      text-indent:........px; }
```

A *text-indent*-el a cím vízszintes helyzetét lehet állítani (az ábra az alapértelmezett állapotot, azaz a *0px*-t mutatja - ekkor nem is kell kódolni a *text-indent*-et):



b) A CSS-kód, ha a címsor az összes hasábot átíveli:
(a HTML-kód és a *div* CSS-kódja változatlan)

```
h2 {
      -webkit-column-span:all;
      -moz-column-span:all;
      column-span:all;
      text-indent:...........px; }
```



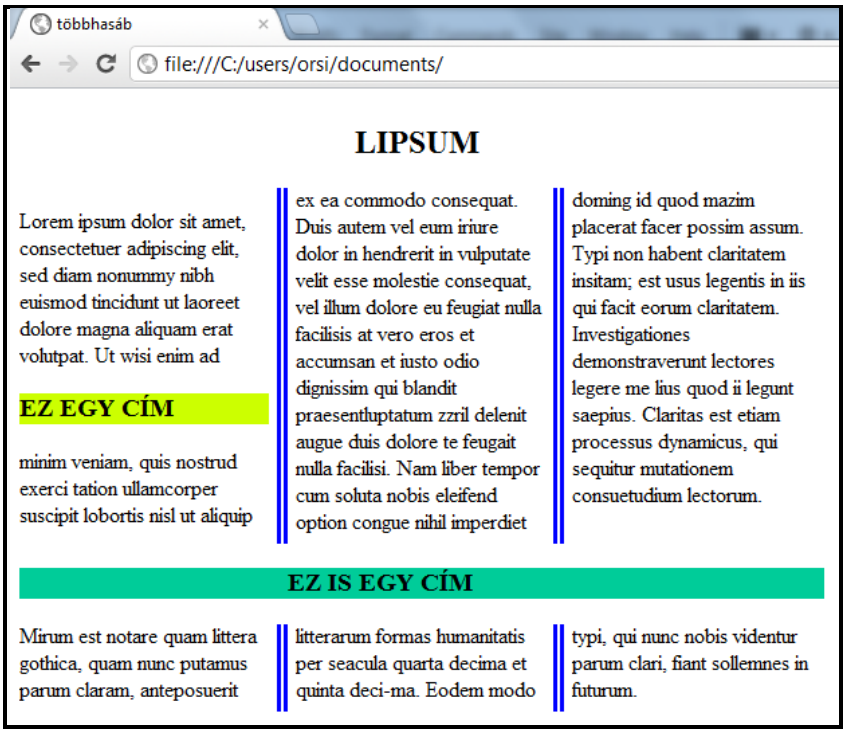
Megjegyzés: A *text-indent:250px* -es helyzetet mutatja az ábra - a cím vízszintes helyzetét szabadon lehet állítani (és a szokásos szöveg formázásokat alkalmazni rá).

Figyelem! A Firefox hibásan jeleníti meg a *column-span:*tulajdonságot!

A címeket a szöveg belsejében is az ismert módon lehet formázni. Ha két *h3* címet beleírunk valahova a szövegbe és „egy” ill. „ketto” azonosítóval látjuk el őket, hogy külön lehessen hivatkozni rájuk, akkor a korábbi CSS-kódunkhoz hozzáírva:

```
#egy {
        -webkit-column-span:none;
        -moz-column-span:none;
        column-span:none;
        background-color:#CF0; }
#ketto {
        -webkit-column-span:all;
        -moz-column-span:all;
        column-span:all;
        background-color:#0C9;
        text-indent: 200px; }
```

A megjelenítés az alábbi lesz:



<u>A többhasábos elrendezéshez felhasznált CSS tulajdonságok:</u>

column-count	hasábok száma
column-width	hasábok szélessége
columns	hasábok számának és szélességüknek összevont alakja
column-gap	hasábköz
column-rule-width	elválasztó vonal szélessége
column-rule-style	elválasztó vonal stílusa
column-rule-color	elválasztó vonal színe
column-rule	elválasztó vonal összevont tulajdonságai
column-span	hasábátfogás

3.26. Színátmenetek

A színátmenet *(gradient*) olyan kép, melyben egy adott szín folyamatosan átmegy egy vagy több másik színbe, és ezt a színhatást a böngésző automatikusan maga hozza létre a CSS kódolás alapján. Háttérképként vagy felsorolásjel képeként egyaránt használható, előállításához nincsen szükség képszerkesztő programra, rugalmasan illeszkedik a tartalom elrendezéséhez, és a valódi képekhez képest sokkal kisebb fájlméret adódik.

A színátmenetnek nincsen saját *(intrinsic*) mérete; ha háttérképnek használjuk, a háttér mindenkori területe, ha felsorolásjel képeként használjuk, akkor a szokásos *1em*-es négyzet lesz a mérete (felsorolásjelben viszont csak egy böngészővel működik).

Két alakzat szerint alakíthatók ki a színátmenetek, egy egyenes vonal mentén (*linear-gradient*) vagy ellipszis alakban (*radial-gradient*). A kör alakú színátmenet az ellipszis azon speciális esete, mikor a vízszintes és függőleges tengely egyenlő.

A színátmenet megadása pl. lineáris háttérképként, ill. sugárirányban változó színű felsorolásjel-képként (ez utóbbi egyelőre csak elvi lehetőség):

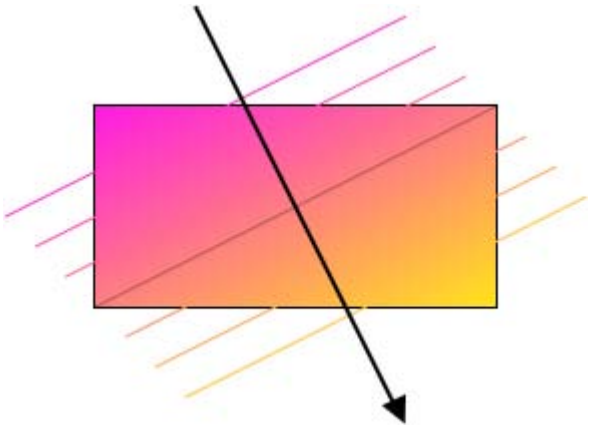
```
background-image: linear-gradient(……);
list-style-image: radial-gradient(……);
```

A zárójelbe az átmenetek paraméterei kerülnek.

Megjegyzés: A szabványos kódolást az Internet Explorer a 10-től értelmezi.

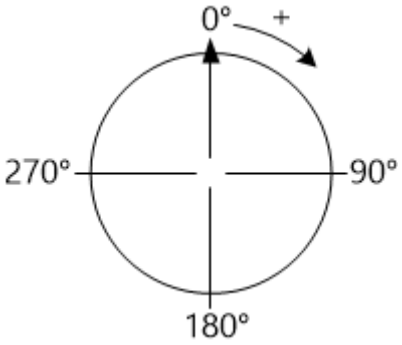
3.26.1. Lineáris színátmenetek

A lineáris színátmenetet egy színátmenet-vonallal (*gradient-line*) és rajta elhelyezett színekkel lehet specifikálni. A gradiens-vonalra merőlegesen elhelyezkedő, egymáshoz illeszkedő vonalak színe megegyezik a színátmenetvonallal való metszéspontjukban lévő színnel, így a színátmenet-vonal színelőírásait követő, végtelen méretű vászon jön létre a színes vonalakból:

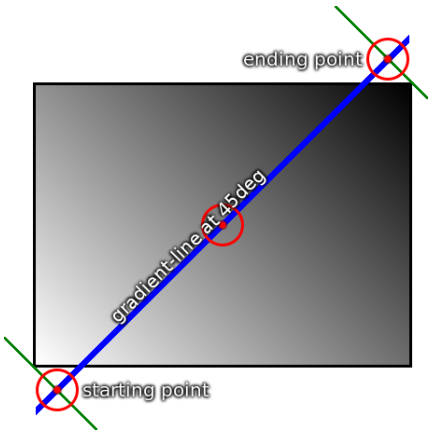


A gradiens-vonal adja meg tehát a színátmenet irányát, és rajta helyezkednek el a színinformációk. A gradiens-vonal mindig áthalad a (háttérkép vagy felsorolásjel kép) doboz közepén - tehát a középpontját nem kell megadni -, iránya és kezdő- ill. végpontja két módon specifikálható:

a) Irányát a függőlegessel bezárt szögével adjuk meg - a 0° felfelé mutat, a 90° fok jobbra, a pozitív szögek az óramutató járásával megegyező irányban mozognak:



Kezdő- és végpontja pedig úgy jön létre, hogy a doboz közepétől mindkét irányban meghosszabbítjuk a gradiens-vonalat a megadott szögben, és a kezdő- és végpont ott van, ahol a vonalra húzott merőleges a doboz sarkát metszi:



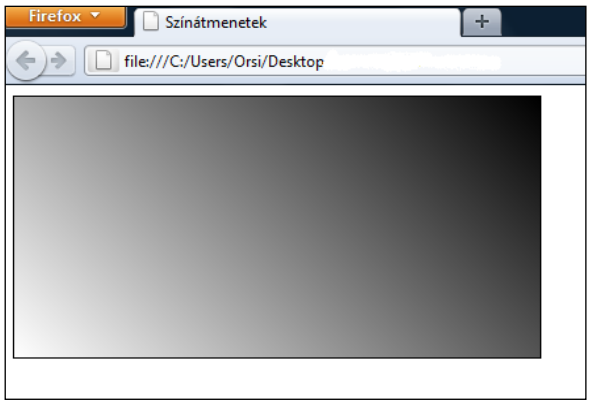
b) A doboznak azt a sarkát vagy oldalát adjuk meg, ahonnan a gradiens-vonal indul, ekkor a gradiens automatikusan az ellentétes sarokba vagy oldalra irányul. (Valójában a gradiens konvertálódik az előző, szögben megadott formulába.) Ha oldalként a *to top*, *to right*, *to bottom* vagy *to left* van megadva, akkor az a 0°, 90°, 180° vagy 270° foknak felel meg. Ha sarok van megadva (*to top left*, *to top right*, *to bottom left*, *to bottom right*), az olyan szögnek felel meg, hogy a középponton áthaladó gradiensvonal a jelzett sarok felé irányuljon.

A gradiens-vonal definiálása után a kezdő- és zárószínt kell megadni, egymástól vesszővel és betűközzel elválasztva. (A színek megadása az előtérszínnél ismertetett módon történik, tehát pl. *red*, vagy *#f00*, vagy *#008000*, stb.)

Az előző, kezdő- és végpontot bemutató ábra hátterének megjelenítéséhez a HTML-kódban egy üres *div*-et alkalmazunk, ennek formázására a CSS-kódolás:

```
<style>
   div {
         background-image:linear-gradient(45deg, white, black);
         width:400px;
         height:200px;
         border:solid 1px black; }
</style>
```

A megjelenítés:



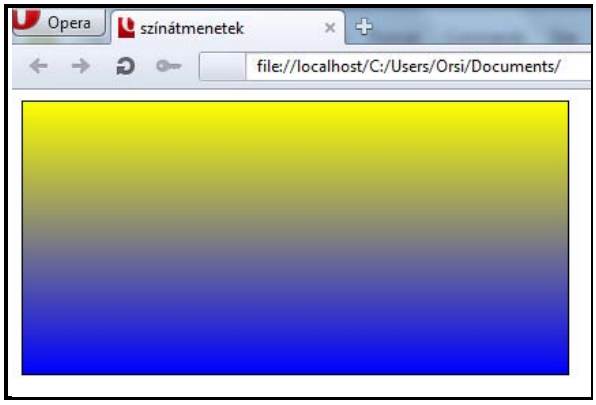
Bár a színátmenet-vonal kezdő- és végpontja a dobozon kívül van, a kezdő és végső szín a doboz sarkokra értendők.

Hasonló átmenetet ad a *linear-gradient(to bottom left, white, black)* kódolás is. Négyzet alakú háttér esetében teljesen megegyező lenne, fekvő téglalapról lévén szó viszont a gradiens-vonal szöge kisebb 45°-nál.

Nézzük meg egy sárgából egyenletesen, függőlegesen, kékbe átmenő háttér lehetséges kódolásait, az előző *div*-re egyenként alkalmazva az egyes kód-sorokat:

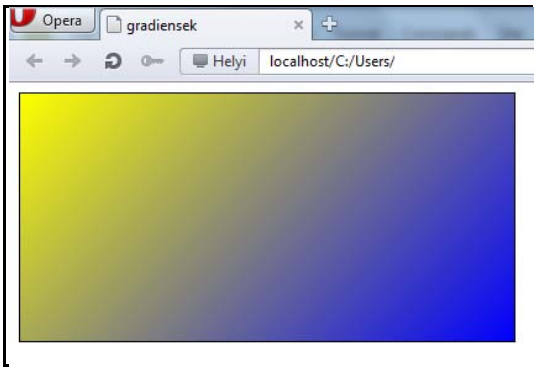
```
background-image: linear-gradient(yellow, blue);
background-image: linear-gradient(to bottom, yellow, blue);
background-image: linear-gradient(to top, blue, yellow);
background-image: linear-gradient(180deg, yellow, blue);
background-image: linear-gradient(to bottom, yellow 0%, blue
                                  100%);
```

Mindegyik ugyanazt a megjelenítést eredményezi. Látható, hogy ha nincsen a gradiens-vonal szöge megadva, akkor fentről lefelé mutató függőleges gradienst alkalmaz (*to bottom*) a böngésző, a színek sorrendja a megadott kiindulási ponttól függ, és a színek helye megadható a gradiens-vonalon (ennek a következőkben, több szín esetén lesz jelentősége):



Ferde gradienst is többféleképpen - pl. a hajlásszögének pozitív vagy negatív értékkel való megadásával - lehet ekvivalens módon definiálni:

```
background-image: linear-gradient(135deg, yellow, blue);
background-image: linear-gradient(-225deg, yellow, blue);
background-image: linear-gradient(-45deg, blue, yellow);
```

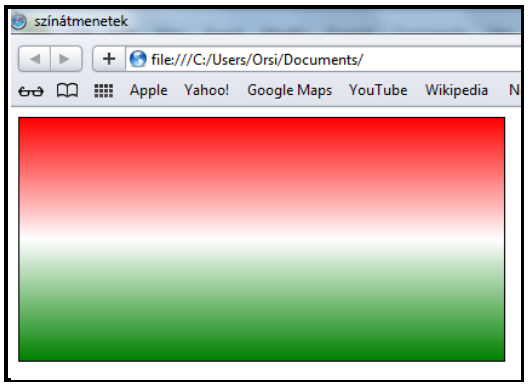


Ha a gradiens-vonal mentén több színátmenet történik, színváltópontokkal *(color-stop)* kell a színátmenetek helyét megadni. A színváltó pontok tipikusan a kezdő- és végpont között helyezkednek el, de ez nem szükségszerű, hiszen a gradiens-vonal mindkét irányban végtelen kiterjedésű. A kezdő- és végpontok csak távolságmarkerek, azt definiálják, hol van a 0%, ill. a 100%, amikhez képest a színváltó pontok helyzetét megadjuk (ami lehet a 0% előtt vagy a 100% után is).

A következő példa 3-színű gradienst mutat be, a *color-stop* pozíciójának megadásával vagy a nélkül. A *color-stop* az érintett színt követi vessző nélkül de szóközzel, helyzete a hosszra alkalmazható mértékegységek felhasználásával (*px*-ben, *%*-ban, *em*-ben, stb.) definiálható.

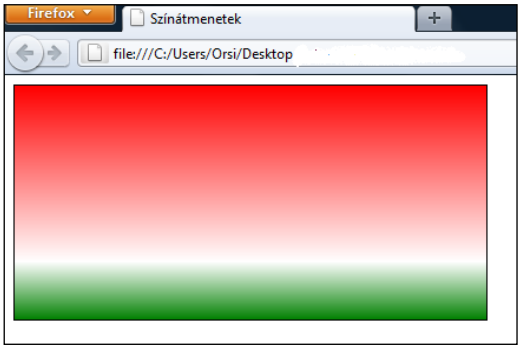
a) ha nincsen a *color-stop* pozíciója megadva, a böngésző egyenletesen osztja el a színeket:

```
background-image: linear-gradient(red, white, green);
```



b) ha definiáljuk a *color-stop* pozícióját, a böngésző a gradiens-vonal hosszát veszi 100%-nak és ennek arányában osztja el a színeket:

```
background-image: linear-gradient(red, white 75%, green);
```



3.26.2. Sugárirányú színátmenetek

A színátmenetek egy pontból kiindulva ellipszis alakban (speciális esete a kör) alakulnak ki. Meg kell adni a gradiens középpontját (ez lesz a 0%-os ellipszis), mely a lineárissal ellentétben tetszőleges helyen lehet, majd specifikálni kell az ellipszis alakját és méretét (ami a 100% lesz), végül a lineáris színátmenethez hasonlóan definiálni a *color-stop*-ok pozícióját. A középpont és a végső ellipszis között koncentrikus ellipszisek alakulnak ki a *color-stop*-okban megadott helyek és az előírt színek szerint.

Az első paraméter az ellipszis középpontját – egyben a gradiens középpontját - definiálja a háttérképeknél leírt *background-position* tulajdonság szerint. Ha nincs megadva, a böngésző *center*-ként értelmezi. A *color-stop*-ok a középtől jobbra induló képzeletbeli vonalon helyezkednek el.

A második paraméter a végső ellipszis alakját és méretét definiálja. Ez két módon adható meg: implicit módon a *shape* (alak) és a *size* (méret) szavakkal, vagy explicit módon.

- a *shape* lehet: *circle* (kör) vagy *ellipse* (ellipszis) – csak vízszintes és függőleges irányú tengelyekkel, ferdével nem !

Ha a *shape* nincsen definiálva, akkor a *size* paraméter alapján dönt a böngésző:

- ha csak egy méret van, akkor kört választ
- ha két méret van, vagy nincsen méret, akkor ellipszist választ

- a *size* lehet (mind *circle*, mind *ellipse* esetén):

- *closest-side* (legközelebbi oldal): a kör érinti a doboznak a gradiens középpontjához legközelebb eső oldalát, az ellipszis a középpontjához legközelebbi függőleges és vízszintes dobozoldalakat érinti. (Ha a középpont a dobozon kívül van, a doboz oldalait meg kell hosszabbítani, hogy az ellipszis érinthesse őket.)

- *farthest-side* (legtávolabbi oldal): a *closets-side*-nak az ellenkezője, azaz a legtávolabbi oldal(ak)hoz igazodnak az alakzatok

- *closest-corner* (legközelebbi sarok): az alakzat mérete úgy alakul, hogy érinti a középpontjához legközelebbi sarkot

- *farthest-corner* (legtávolabbi sarok): az alakzat mérete úgy alakul, hogy érinti a középpontjához legtávolabbi sarkot

Ha a *size* nincsen definiálva, akkor a *farthest-corner* –t választja a böngésző.

- explicit módon a végső alak mérete és alakja két hosszmérettel vagy két százalékos értékkel adható meg - ezek az ellipszis vízszintes és függőleges tengelyeinek a hosszát (az ellipszis középpontjától a széléig, tehát pl. kör esetében a sugárt, nem az átmérőt) jelentik. Százalékos megadás esetén az első érték a doboz szélességéhez, a második érték a doboz magasságához viszonyítást jelenti. Csak nullánál nagyobb (pozitív) érték adható meg.

A gradiens-vonal végpontja a kezdőpontból a megadott szögben meghúzott egyenesnek a végső ellipszissel alkotott metszéspontja.

Bizonyos esetekben a megadott paraméterek degenerált alakot – egy 0 sugarú ellipszist vagy kört - eredményezhetnek. Ekkor a színátmenet csak egy szín, mely megegyezik a gradiens-vonalon lévő utolsó *color-stop* színével. A következő értékkombinációknál fordulhat ez elő:

- *closest-side* ha a kezdőpont a doboz szélén van
- *closest-corner* ha a kezdőpont a doboz sarkán van
- *ellipse* és *closest-corner* ha a kezdőpont a doboz szélén van.

Ha a középpontot, alakot és/vagy méretet definiáljuk, először az alak, aztán a méret, majd a középpont következik *at* szócskával összekötve, pl.: *ellipse at center, farthest-corner at 50% 50%, closest-side at 40px 60px, farthest-side at left bottom, 40px 60px at 40px 60px,* stb. (Ezek az értékek a továbbiakban a bemutató példákban konkrétan előfordulnak, tartalmukat ott demonstráljuk.)

A *color-stop*-ok a lineáris színátmenetnél látottakkal analóg módon működnek. Az átmenet középpontjából húzott képzeletbeli vonalon helyezkednek el, a 0%-os pont van a középpontban, és a 100%-os pont a vonal és a végső ellipszis metszéspontjában. 100%-nál nagyobb érték is megadható, és negatív hossz is lehetséges (de ekkor az átmenetből a kezdőpont előtti színek nem látszanak).

A fentiek illusztrálására szolgáló bemutató példák következnek:

a) Középről induló, két színű, a doboz széléig nyúló ellipszis alakú egyenletes színátmenet:

```
background-image:radial-gradient(yellow, green);
background-image:radial-gradient(ellipse at center, yellow 0%,
                                 green 100%);
background-image:radial-gradient(farthest-corner at 50% 50%,
                                 yellow, green);
```

Ezek ekvivalens kódolások, mert:

- a *position* megadása *center* kulcsszóval, *50-50 százalékosan*, vagy elhagyása esetén a böngésző automatikusan a közepet választja

- az *alak* megadása *ellipse* kulcsszóval, vagy elhagyása esetén a méret hiánya, ill. téglalap alakú doboznál a *farthest-corner* két méret meglétére való utalása miatt a böngésző automatikusan az ellipszist választja

- a *color-stop* értékek megadása vagy elhagyása jelen esetben nem jelent plusz információt, mivel kezdő- és végpontról van szó

-a *size* megadása ha elmarad, a böngésző automatikusan a *farthest-corner*-t választja

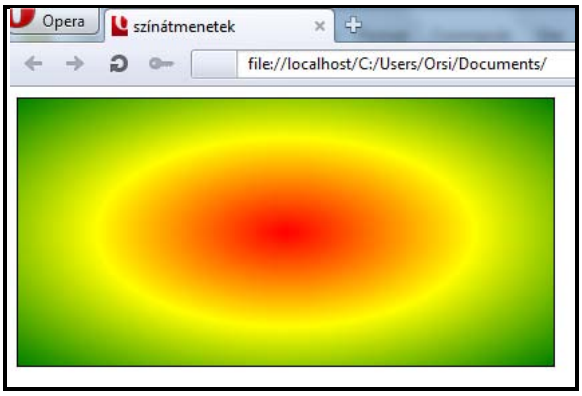
b) Középről induló, két színű, a doboz széléig nyúló ellipszis alakú színátmenet *color-stop*-okkal (a példa kedvéért az első színre a kezdőponthoz képest negatív hosszt megadva):

```
background-image:radial-gradient(yellow -100px, green 200px);
```

A gradiens-vonal kezdőpontja előtti szín (sárga) nem látszik, a kezdőpontnál lévő szín-átmeneti fázisból (sárgás-zöldesből) indul a látható színátmenet a zöldbe.

c) Középről induló, három színű, doboz széléig nyúló ellipszis alakú egyenletes színátmenet:

```
background-image:radial-gradient(red, yellow, green);      vagy pl.
background-image:radial-gradient(farthest-corner at 50% 50%,
                                 red, yellow, green);
```



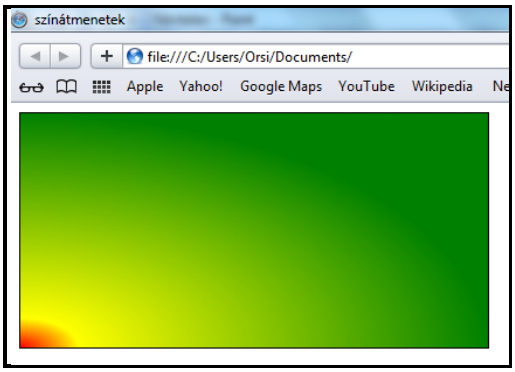
d) Nem középről induló, adott méretű, három színű, egyenletes színátmenet:

```
background-image:radial-gradient(closest-side at 40px 60px,
                                 red, yellow, green);
background-image:radial-gradient(40px 60px at 40px 50px, red,
                                 yellow, green);
```

A két kódolás azért ekvivalens, mert az ellipszis féltengely-méreteinek pont a legközelebbi oldalaktól való távolságot adtuk meg.

e) A doboz szélén vagy azon kívül is lehet a gradiens-vonal kezdőpontja, és legyen nem egyenletes a színátmenet:

```
background-image:radial-gradient(farthest-side at left bottom,
                                 red, yellow 50px, green);
```



3.26.3. Periódikusan ismétlődő színátmenetek

A színátmenet nem csak egyszeri lehet, hanem periodikusan ismétlődhet mind egyenes vonal mentén (*repeating-linear-gradient*), mind ellipszis alakú színátmenetek esetén (*repeating-radial-gradient*).

A periódikusan ismétlődő színátmenet megadása lineáris ill. sugárirányban változó színű háttérképként:

```
background-image: repeating-linear-gradient(................);
background-image: repeating-radial-gradient(................);
```

A színátmenetek értelmezése és értékei megegyeznek a korábban ismertetett nem periodikus átmenetekével.

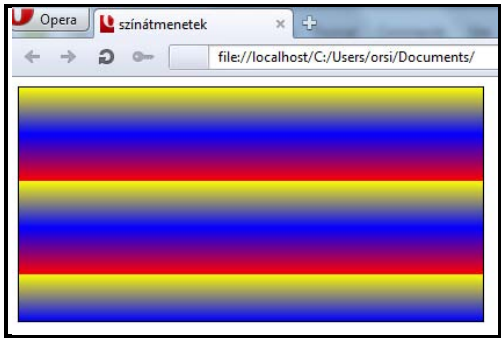
A *color-stop*-ok mindkét irányban végtelenül ismétlődnek az utolsó *color-stop* és első *color-stop* helyzete különbségének többszörösével eltolt pozícióval. Pl. a ***repeating-linear-gradient(****red 10px, blue 50px****)*** érték megegyezik a ***linear-gradient(****..., red -30px, blue 10px, red 10px, blue 50px, red 50px, blue 90px, ...****)*** értékkel.

Az első és utolsó *color-stop* mindig egybeesik az ismétlődő átmenetcsoportok határán, ami éles átmeneteket okozhat, ha a színátmenet nem ugyanazzal a színnel indul és fejeződik be.

Példák a periódikusan ismétlődő színátmenetekre:

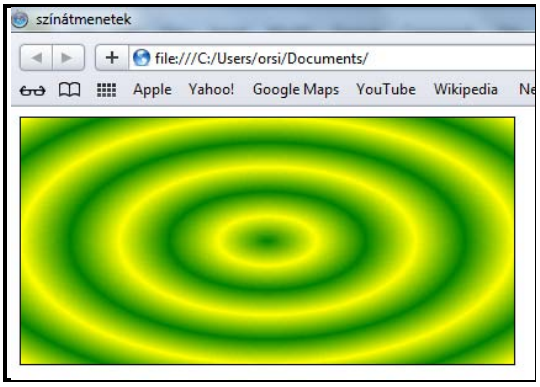
a) Lineáris periodikus átmenet, a kezdő és végső szín nem azonos:

```
background-image:repeating-linear-gradient(yellow, blue 40px,
                                           red 80px);
```



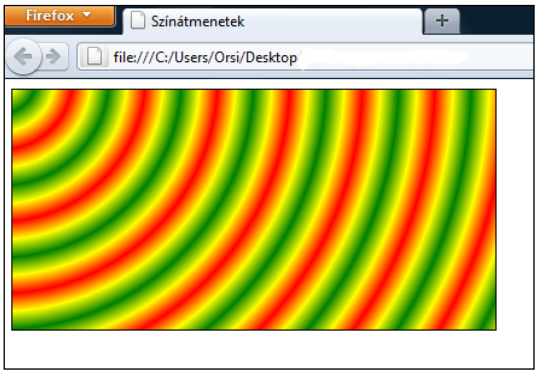
b) Középről induló, elliptikus, periódikus átmenet, a kezdő és végső szín azonos:

```
background-image:repeating-radial-gradient(green, yellow 40px,
                                           green 80px);
```



c) Nem középről, sőt dobozon kívülről induló periodikus kör alakú, három színű átmenet:

```
background-image:repeating-radial-gradient(circle closest-side
         at -10px -10px, red, yellow, green 300%, yellow 450%,
         red 600%);
```



A színátmenetek várhatóan legfontosabb alkalmazási területei a weblapok különböző gombjainak és a geometrikus háttérképeknek a kialakítása lesz.

a) A *Standard vonaltípusú lekerekített szegélyek* és a *Doboz árnyékok* fejezetekben már alakítottunk ki formázott gombokat, ennek a folytatásaként most színátmenetekkel is formázunk. Kiindulás legyen egy lekerekített sarkú, szegély nélküli, egyszínű zöld háttérrel rendelkező gomb a korábbiak szerint:

```
button { width:155px;
         height:35px;
         text-align:center;
         vertical-align:middle;
         font-family:Corsiva, serif;
         color:yellow;
         font-weight:bold;
         background-color:green;
         border-radius:20px;
         font-size:24px; }
```

A HTML-dokumentum törzse pedig egyszerűen: `<button>GOMB</button>`

A két leggyakoribb színátmenetes megoldás az alulról felfelé világosodó gomb, ill. a középről lefelé és felfelé sötétedő gomb.

A fenti CSS-kódba beírva az alulról felfelé világosodásó zöldet:

```
button { width:155px;
         height:35px;
         text-align:center;
         vertical-align:middle;
         font-family:Corsiva, serif;
         color:yellow;
         font-weight:bold;
         background-color:green;
         border-radius:20px;
         font-size:24px;
         background-image:linear-gradient(to top, #060, #0f0); }
```

vagy még egyszerűbben:

```
         background-image:linear-gradient(#0f0, #060);
```

A fenti CSS-kódban megváltoztatva a színátmenet jellegét a középen világos verzióra, és megszüntetve az alapértelmezett szegélyárnyékot:

```
button { width:155px;
         height:35px;
         text-align:center;
         vertical-align:middle;
         font-family:Corsiva, serif;
         color:yellow;
         font-weight:bold;
         background-color:green;
         border-radius:20px; font-size:24px;
         background-image:linear-gradient(#060, #0f0, #060);
         border-width:0; }
```

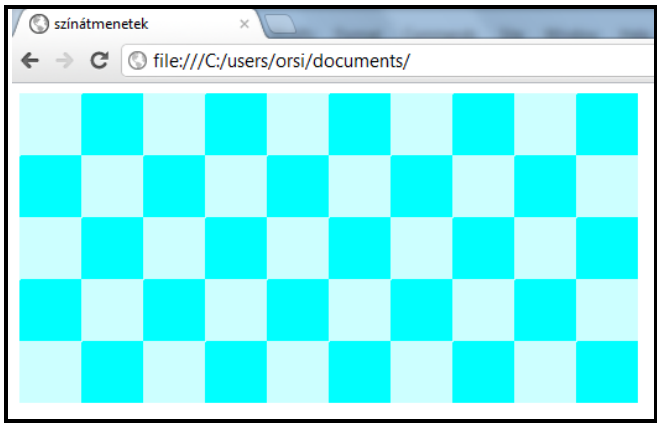
Azok a böngészők, melyek nem értelmezik a színátmenetet és a lekerekített szegélyt, egyszerűen zöld hátteret és szögletes gombot jelenítenek meg (lásd alul a jobb oldali gomb), a többiek felülírják a sorban előttük elhelyezkedő háttérszínt, és a kódolt színátmenetet mutatják.

A *button* címkéhez alapértelmezetten adnak a böngészők némi szegélyt, ha nem tartunk rá igényt, kódolással lehet megszüntetni (0 szegélyvastagság, lásd a második formázási kódsort, alul a középső gombot) vagy helyette másmilyen szegélyt kódolni.

b) Többszörös háttérképekkel és gradiensekkel változatos, periódikusan ismétlődő geometrikus alakzatú hátterek alakíthatóak ki képek alkalmazása nélkül (IE csak a 10-től):

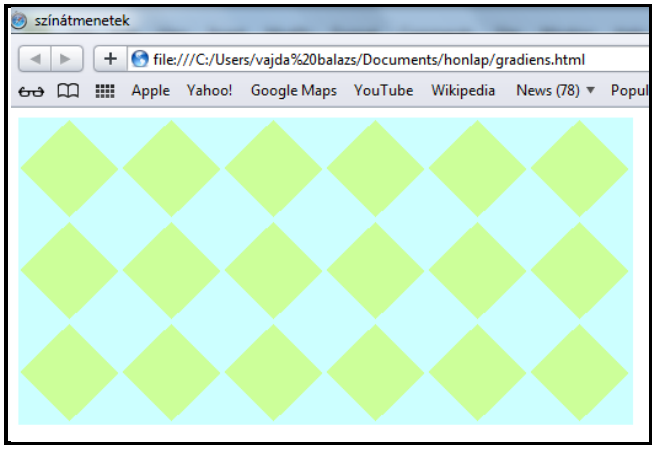
Néhány példa CSS-kódjának bemutatása *500px* x *250px*-es *div*-en (a HTML-kód minden esetben ez az üres *div*):

a)
```
div {
    width:500px; height:250px;
    background-color:#CFF;
    background-size:100px 100px;
    background-position:0 0, 50px 50px;
    background-image:linear-gradient(45deg, aqua 25%,
    transparent 25%, transparent 75%, aqua 75%, aqua),
    linear-gradient(45deg, aqua 25%, transparent 25%,
    transparent 75%, aqua 75%, aqua); }
```



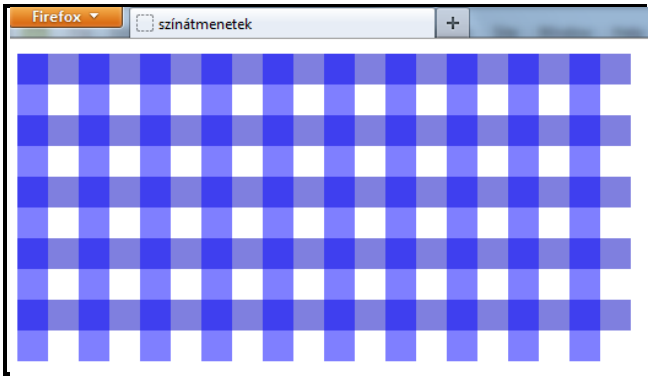
A sakktábla-szerű háttér kockáinak méretét a *background-size*-al lehet állítani, a *background-position* második értéke a *background-size* mindenkori értékének a fele kell hogy legyen.

b)
```
div {
    width:500px; height:250px;
    background-color:#CF9;
    background-size:83px 83px;
    background-image:linear-gradient(45deg, #CFF 25%,
    transparent 25%, transparent 75%, #CFF 75%, #CFF),
    linear-gradient(-45deg, #CFF 25%, transparent 25%,
    transparent 75%, #CFF 75%, #CFF); }
```



A kárómintа-szerű háttér kockáinak méretét szintén a *background-size*-al lehet állítani.

c)
```
div {
    width:500px; height:250px;
    background-color:white;
    background-size:50px 50px;
    background-image:linear-gradient(180deg, 0, rgba(0%, 0%,
    100%, .5) 50%, transparent 50%), linear-gradient(180deg,
    rgba(0%, 0%, 75%, .5) 50%, transparent 50%); }
```



Az asztalterítő minta-szerű háttérnél is a *background-size*-al állítható a minta sűrűsége.

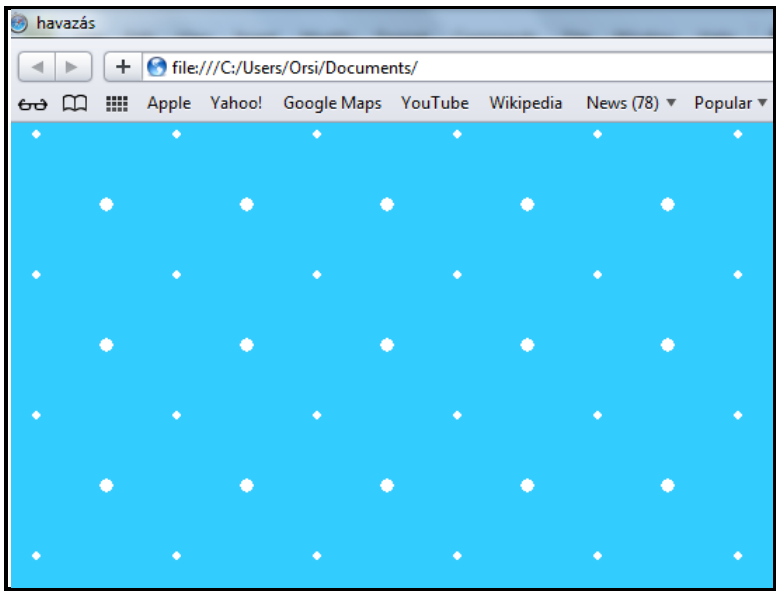
d)
```
div {
    width:500px; height:250px;
    background-color:red;
    background-size:60px 60px;
    background-position:0 0, 30px 30px;
    background-image:radial-gradient(white 25%, transparent
    25%), radial-gradient(white 25%, transparent 25%); }
```

A pöttyös háttérmintánál a pöttyök mérete a %-os értékkel, a köztük lévő vízszintes és függőleges távolság a *background-size*-al állítható - a *background-position* második értéke a *background-size* mindenkori értékének a fele kell hogy legyen. A pöttyök mérete páros/ páratlan soronként eltérő méretűre kódolható – az egyik sort az egyik háttérkép %-a, a másik sort a másik háttérkép %-a jelenti. Az egyenletes rácsszerkezetet torzítani a *background-position* és *background-size* független változtatásával lehet.

A következő háttérformázást animációval kombinálva például havazás imitálható. Mivel a törzsre (*body*) alkalmazzuk a tulajdonságokat, a *div*-re sincs a HTML-kódban szükség. A pöttyök soronként kisebbek/nagyobbak, a változás állandó sebességű és végtelenített, az esés szöge enyhén eltér a függőlegestől (és az animáció jelenlegi támogatottsága miatt a –*webkit*-es kódot is bele kell írni):

```
<style>
   body {
         background-image:radial-gradient(white 7%, transparent
                 7%), radial-gradient(white 4%, transparent 4%);
         background-color:#3CF;
         background-size:100px 100px, 100px 100px;
         background-position:0px 0px, 50px 50px;
         -webkit-animation:havazas 60s linear infinite;
         animation:havazas 60s linear infinite; }
         @-webkit-keyframes havazas {
             0% { background-position:0px 0px, 50px 50px; }
           100% { background-position:30px 150px, 80px 200px; }
           }
         @keyframes havazas {
             0% { background-position:0px 0px, 50px 50px; }
           100% { background-position:300px 1500px, 350px
                                        1550px; }
          }
</style>
```

Azért adtunk meg hosszú animációs időt és utat, hogy a ciklus végén előforduló rángatás ritkán jöjjön elő. A két pötty mozgási dőlésszögét egyformára érdemes állítani (0% és 100% -os állapotok koordinátáinak a különbsége megegyezzen), hogy párhuzamosan hulljanak a közeli (nagyobb) és távoli (kisebb) pelyhek:



Egy harmadik hópiheméretű háttérkép bekódolásával, a hátterek helyzetének aszimmetrikus beállításával, és az egyes háttereknek enyhén eltérő sebességet adva a hatás tovább javítható.

<u>A színátmenetekhez felhasznált CSS tulajdonságok:</u>

linear-gradient(……..)	lineáris színátmenet
radial-gradient(……..)	sugárirányú színátmenet
repeating-linear-gradient(…….)	periódikusan ismétlődő lineáris színátmenet
repeating-radial-gradient(…….)	periódikusan ismétlődő sugárirányú színátmenet

3.27. Síkbeli transzformációk

Síkbeli transzformáció (*2D transform*) esetén a vízszintes *(x)* és/vagy függőleges *(y)* tengely mentén egy elemen eltolás, elforgatás, nagyítás/kicsinyítés vagy elferdítés hajtható végre.

A *transform* (átalakít) tulajdonság értékei (a transzformációs függvények, ill. egyszerűbben fogalmazva az átalakítás jellege) definiálják, hogy milyen művelet végzendő el, a zárójelben megadott szám(ok) pedig az átalakítás mértékét adja(k) meg.

Figyelem! A síkbeli transzformációkat a Firefox, Opera, az Internet Explorer a 10-től, a Chrome (*prefix* nélkül) a 36-tól, a Safari-k és mobil böngészők viszont csak *–webkit-* előtaggal értelmezik, a kódolást emiatt *–webkit-* el is el kell végezni.

A transzformációk hatását legszemléletesebb egy *:hover*-el kombinálva bemutatni, ezért minden műveletet egy *div* eredeti és átalakított állapotának egyidejű kódolásával demonstrálunk. A HTML-kód mindig

```
<div>TRANSZFORMÁCIÓK</div>
```
lesz.

A *transform* tulajdonság lehetséges értékei:

a) A **translate** (áthelyezés, eltolás) egy elem középpontját a megadott *x* és/vagy *y* irányba tolja. Kódolása:

- *translateX(…)* - eltolás vízszintesen (az *x* tengely mentén), pozitív szám esetén jobbra, negatív szám esetén balra (*px*-ben megadva)
- *translateY(…)*..- eltolás függőlegesen (az *y* tengely mentén), pozitív szám esetén lefelé, negatív szám esetén felfelé (*px*-ben megadva)
- *translate(…...)* – eltolás mind vízszintes, mind függőleges irányba; az első adat a vízszintes, a második a függőleges eltolás mértékét jelenti (*px*-ben megadva). Vessző és egy szóköz választja el a két adatot egymástól.

Az eredeti és egy *80px*-el vízszintesen és függőlegesen eltolt *div* CSS-kódja:

```
<style>
     div { width:200px; height:200px; background-color:#3CF;
           text-align:center; }
     div:hover {
           -webkit-transform:translate(80px, 80px);
           transform:translate(80px, 80px); }
</style>
```

b) A **scale** (átméretezés) egy elem középpontjából kiindulva a megadott *x* és/vagy *y* irányba megváltoztatja a méretet. Kódja:

- *scaleX(….)* - átméretezés (nagyítás vagy kicsinyítés) vízszintes irányban
- *scaleY(….)* - átméretezés (nagyítás vagy kicsinyítés) függőleges irányban
- *scale(……)* - átméretezés mind vízszintes, mind függőleges irányban; az első adat a vízszintes, a második a függőleges átméretezés mértékét adja meg. Vessző és egy szóköz választja el a két adatot egymástól.

Az eredeti és egy, a középponttól számítva mindkét tengely irányában 50%-al megnövelt méretű *div* CSS-kódja:

```
<style>
      div { width:200px; height:200px; background-color:#3CF;
            text-align:center; }
      div:hover {
      -webkit-transform:scale(1.5, 1.5);
      transform:scale(1.5, 1.5); }
</style>
```

Megjegyzés: A *transform:scaleX(-1)* az elem vízszintes tengelyére tükrözést, a *transform:scale Y(-1)* a függőleges tengelyére tükrözést, a *transform:scale(-1, -1)* a mindkét tengelyére tükrözést valósítja meg.

c) A **rotate** (elforgatás) egy elem középpontja körül adott szöggel való elforgatást hajt végre. Kódja:

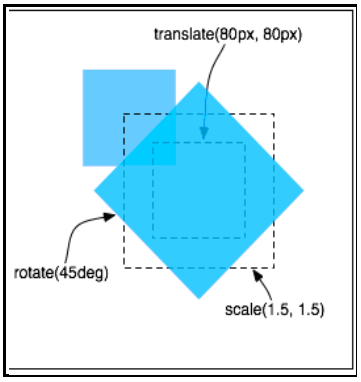
- *rotate(….)* - adott szöggel (*deg*-ben megadva) való elforgatás az elem origója körül; pozitív érték az óramutató járásával megegyező irányt jelöl

```
<style>
      div { width:200px; height:200px; background-color:#3CF;
            text-align:center; }
      div:hover {
      -webkit-transform:rotate(45deg);
      transform:rotate(45deg); }
</style>
```

Egy elemen egyidejűleg több transzformációs művelet is végrehajtható. Ilyen esetben összevont alakban, vessző nélkül, a műveletek sorrendjében felsorolva adandók meg a transzformációs függvények és értékeik. Az eddig bemutatott három művelet egyidejű megvalósításának kódja tehát:

```
<style>
      div { width:200px; height:200px; background-color:#3CF;
            text-align:center; }
      div:hover {
            -webkit-transform:translate(80px, 80px) scale(1.5,
                                 1.5) rotate(45deg);
            transform:translate(80px, 80px) scale(1.5, 1.5)
                        rotate(45deg); }
</style>
```

A *div* kezdeti és az egyes transzformációkat bemutató állapota:



d) A **skew** (elferdítés) a megadott *x* és/vagy *y* irányban ferdén megdönti az elemet. Kódja:

- *skewX(….)* - elferdítés a vízszintes tengely mentén (*deg*-ben megadva)

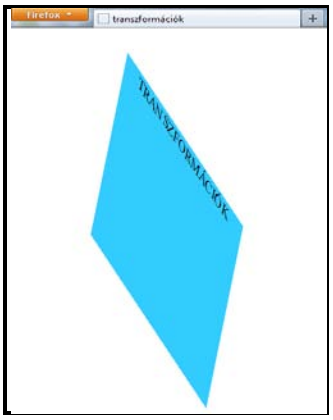
- *skewY(….)* - elferdítés a függőleges tengely mentén (*deg*-ben megadva)

- *skew(…....)* - elferdítés mind a vízszintes, mind a függőleges tengely mentén; az első adat a vízszintes, a második a függőleges elferdítés mértékét (*deg*-ben megadva) jelenti. Vessző és egy szóköz választja el a két adatot egymástól.

A korábbi transzformációkkal egyidejűleg egy 20° vízszintes és 30° elferdítést is végrehajtva a CSS-kód:

```
<style>
      div { width:200px; height:200px; background-color:#3CF;
            text-align:center; margin-top:100px; }
      div:hover {
            -webkit-transform:translate(80px, 80px) scale(1.2,
                        1.2) rotate(30deg) skew(20deg, 30deg);
            transform:translate(80px, 80px) scale(1.2, 1.2)
                        rotate(30deg) skew(20deg, 30deg); }
</style>
```

A böngészőkben a megjelenítés:



Amennyiben a transzformáció origo-ja nem esik egybe a transzformálandó elem középpontjával, a *transform-origin* tulajdonsággal adható meg. (Az eddigi példák mind az alapértelmezetten az elem közepére vonatkoztak.) Kódja:

- *transform-origin(……)* - a transzformáció origo-jának helyzete az elem bal felső sarkához képest az *x* és *y* tengely irányában. Megadható százalékosan (az 50% 50% az elem közepe) *px*-ben, vagy szavakkal. Az x-tengely esetén a *left, center* és *right*, az y-tengely esetén a *top, center* és *bottom* szavak használhatók. Az első érték mindíg a vízszintes, a második a függőleges adatot jelenti, kettőjük között egy üres szóköz van.

Ha pl. a *div*-et a jobb alsó sarka körül forgatjuk el az óramutató járása szerinti 45°- al, akkor a kurzor-hatás kódolása:

```
div:hover {
            -webkit-transform:rotate(45deg);
            -webkit-transform-origin:right bottom;
            transform:rotate(45deg);
            transform-origin:right bottom; }
```

Ha pl. a *div*-et úgy nagyítjuk fel, hogy a bal oldala és felső éle változatlanul a helyén maradjon (azaz csak jobbra és lefelé terjeszkedjen a nagyítás hatására), akkor a kurzor-hatás kódolása:

```
div:hover {
            -webkit-transform:scale(1.5, 1.5);
            -webkit-transform-origin:left top;
            transform:scale(1.5, 1.5);
            transform-origin:left top; }
```

A *transform-origin* a transzformálandó elemen kívül is elhelyezkedhet, pl. a *div*-et egy tőle *100px*-el jobbra eső függőleges tengelyre tükrözve a kódolás:

```
div:hover {
            -webkit-transform:scaleX(-1);
            -webkit-transform-origin:300px;
            transform:scaleX(-1);
            transform-origin:300px; }
```

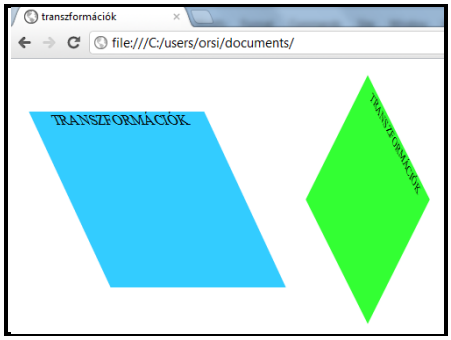
Transzformációkkal (CSS-kódolással és nem képpel) néhány geometriai alakzat kialakítható, melyre példák (a HTML-kód továbbra is az egyetlen *div*-ből áll):

a) paralelogramma

```
div { width:200px; height:200px;
      background-color:#3CF; text-align:center;
      -webkit-transform:skew(25deg);
      transform:skew(25deg); }
```

b) gyémánt

```
div { width:200px; height:200px;
      background-color:#3F3; text-align:center;
      -webkit-transform:scaleX(0.5) rotate(45deg);
      transform:scaleX(0.5) rotate(45deg); }
```



Mint látható, a fenti alakzatokba tartalom is elhelyezhető, és a tartalom követi a CSS-ben az alakzatra kódolt utasításokat.

Transzformációkkal vizuálisan ki lehet emelni elemeket a környezetükből. Egy tipikus példa erre a képek halmazából egy kiválasztott képnek a kinagyítása. Példánkban három képet úgy kódolunk, hogy egymás mellett helyezkedjenek el közepes méretben, majd az egérrel bármelyikük fölé állva a kiválasztott kép 50%-al nagyobb legyen. A látvány kedvéért ilyenkor szegély és doboz-árnyék is társuljon hozzá, és egy kicsit el is forgatjuk a szélső képeket, hogy

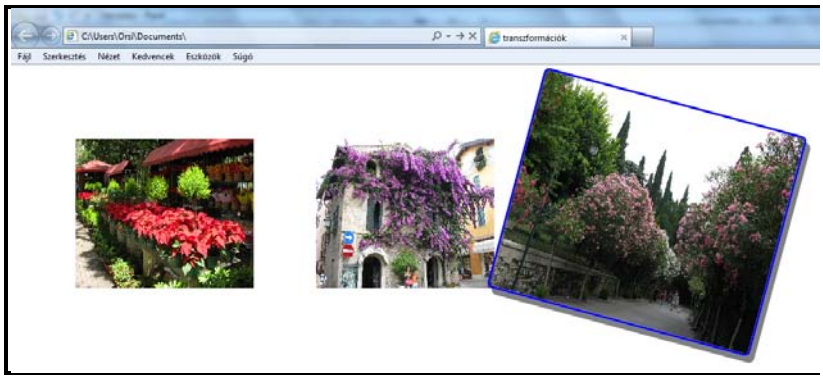
enyhén közép felé forduljanak - a középső nem fordul semerre. Pirossal jelöltek a transzformációk, a kódolást –*prefix*- nélkül írtuk le (de –*webkit*-el is szükséges kiegészíteni):

```
<style>
    li { display:inline; list-style:none; margin-right:80px; }
    ul { padding:80px; }
   #egy:hover { border:2px solid blue; border-radius:5px; box-
                shadow:5px 5px gray; transform:scale(1.5, 1.5)
                rotate(-15deg); }
   #ketto:hover { border:2px solid blue; border-radius:5px;
                 box-shadow:5px 5px gray; transform:scale(1.5,
                 1.5) rotate(0deg); }
         #harom:hover { border:2px solid blue; border-
                        radius: 5px; box-shadow:5px 5px gray;
                       transform:scale(1.5, 1.5) rotate(15deg); }
</style>
```

A HTML-kód (a három kép listában elhelyezve):

```
<ul>
    <li><img id="egy" src="IMG_1010.JPG" width="240px"
         height="200px"></li>
    <li><img id="ketto" src="IMG_1315.JPG" width="240px"
         height="200px"></li>
    <li><img id="harom" src="IMG_1342.JPG" width="240px"
         height="200px"></li>
</ul>
```

A megjelenítés az egérkurzorral a jobb oldali kép fölé állva:



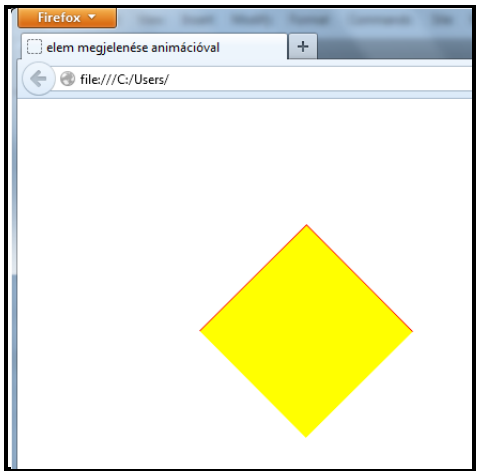
Transzformáció és animáció együttes alkalmazásával egy elem pörögve behozatala vagy pörögve eltüntetése is megvalósítható. Példa egy *div* CSS-el történő pörögve behozatalára – a HTML kód csak egy üres, sárga *div*-et tartalmaz:

```
<style>
       div { width:300px; height:300px; border:solid red 1px;
             background-color:yellow; position:absolute;
             left:10%; top:10%; -webkit-animation:beporog 2s;
             animation:beporog 2s; }
             @-webkit-keyframes beporog {
                 from { transform:scale(0) rotate(1440deg); }
             }
             @keyframes beporog {
                 from { transform:scale(0) rotate(1440deg); }
             }
</style>
```

Az első meghatározás a *div* helyét és végső formáját definiálja. A pirossal jelzett animációval

- a *div* átméretezését 0-ról 1-re, azaz a láthatatlanról a névleges méretére változtatjuk (a végső állapotot nem is kellett kiírni, mert a *scale(1)* az alapértelmezett érték)
- a *div*-et a központja körül (transform-origin: 50% 50%) négyszer megforgatjuk (4 x 360deg = 1440deg) – a központ körüli forgatás az alapértelmezett, tehát nem szükséges külön kiírni
- az animáció (a pörögve behozatal) *2 sec* alatt, ismétlődés és késleltetés nélkül, az alapértelmezett *ease* időbeni lefolyással zajlik le
- a -*webkit*- kódolás hozzáadása a Chrome, Opera és Safari animációs, valamint a Safari transzformációs értelmezése miatt szükséges

A fenti kódolással a weblap megjelenésekor automatikusan megjelenik pörögve a *div*.



Megjegyzés: Ha a *from*-ot *to*-ra írjuk át és *animation-fill-mode:forwards;*-et specifikálunk, akkor automatikus pörögve eltűnés következik be.

Amennyiben valamilyen egér-eseményhez kívánjuk kötni az eltűnést vagy megjelenést, a korábbiaknak megfelelően további kódkiegészítés szükséges.

<u>A síkbeli transzformációkhoz felhasznált CSS tulajdonságok:</u>

translateX(…..)	vízszintes eltolás
translateY(…..)	függőleges eltolás
translate(…….)	síkbeli kétirányú eltolás
scaleX(……)	vízszintes átméretezés
scaleY(…….)	függőleges átméretezés
scale(……..)	síkbeli kétirányú átméretezés
rotate(……..)	elforgatás
skewX(…….)	vízszintes elferdítés
skewY(…….)	függőleges elferdítés
skew(……)	síkbeli kétirányú elferdítés
transform-origin(…….)	transzformáció origo-jának helyzete

3.28. Térbeli transzformációk

Mindjárt az elején fontos leszögezni, hogy bár a *transform* tulajdonság néhány értéke (transzformációs függvénye) lehetővé teszi elemeknek 3-dimenziós koordinátarendszerben történő megváltoztatását, maga az elem nem 3-dimenziós tárgy, hanem csak 2-dimenziós síkon létezik és nincsen mélysége. A perspektívával olyan pontot hozunk létre, mely a pozitív (nézőhöz közelebbi) *z*-koordináták *x* és *y* – értékeit felnagyítja, a negatív (távolabbi) *z*-koordináták *x* és *y* – értékeit lekicsinyíti, ilyen módon előállítva a mélységnek a látszatát.

Megjegyzés: A térbeli transzformációt jelenleg a Firefox, Opera, az Internet Explorer a 10-től, a Chrome a 36-tól, a Safari és a mobil böngészők viszont csak *–webkit-* előtaggal értelmezik, az Internet Explorer-ek pedig nem jelenítik meg a *transform-style:preserve-3d* tulajdonság/érték párost (lásd később).

A térbeli transzformációkat egy *div*-ben elhelyezett második *div*-en mutatjuk be – a befoglaló *div* adja majd a 3-dimenziós teret, melyben a transzformálandó *div*-en hajtjuk végre a „térbeli” műveleteket.

A kiinduló állapot kódolása és megjelenítése:

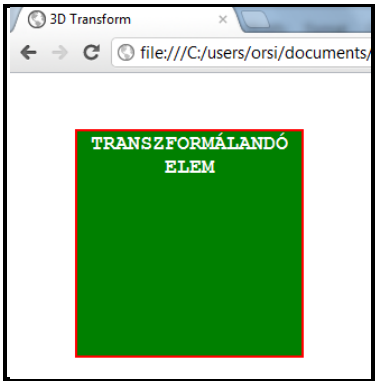
A HTML-kód (ez a későbbiekben végig változatlan marad):

```
<div id="befoglalo">
     <div id="transzformalando">TRANSZFORMÁLANDÓ ELEM</div>
</div>
```

A CSS-kód (a *befoglaló div*-nek csak a szegélye látszik, egyébként takarja a *transzformálandó div*):

```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; }
#transzformalando { background-color:green; color:white;
                    font-size:18px; font-family:"Courier New",
                     Courier, monospace; font-weight:bold;
                     text-align:center; }
```

A kiinduló állapot megjelenítése:



A térbeli látásmódhoz ki kell lépni az *x-y* tengelyek alkotta síkból, és a rájuk merőleges *z*-tengelyen mért távolságból szemlélni az elemet. A nézőnek a *z=0* ponttól mért távolságát a *perspective* (távlat, perspektíva) tulajdonság *px*-ben megadott értéke definiálja (%-os érték

nem értelmezett). Ha negatív szám, *0* vagy *none* az értéke, akkor nincs térbeliség (az elem mögé kerülni nem lehet, ill. benne maradtunk az elem síkjában). Ha kicsit távolodunk el, mindent közelről, hatalmasnak látunk, és minél nagyobb az elem síkjától mért távolság, annál kisebb a változtatások látszólagos hatása.

Nem szükségszerű, hogy a nézőpont rajta legyen a *z*-tengelyen (azaz az elem bal felső sarka felől nézzük az elemet), attól bármelyik irányban eltérő helyre is pozícionálható. A *z*-tengelyhez képesti eltérést a *perspective-origin* tulajdonság értékével, *x* és *y* koordinátákkal (*px*-ben, %-ban, kulcsszavakkal) lehet megadni. Az alapértelmezett a *z*-tengelyen lévő pozíció, mely a (*0px 0px*), ill.(*50% 50%*), ill. (*center center*) koordinátáknak felel meg – ezt nem szükséges kódolni.

Fentiek alapján helyezzük a nézőpontunkat a transzformálandó elem síkjától *600px* távolságra (ez az elem méretéhez képest közepes távlatnak tekinthető), és a bal felső sarkától kicsit (*50px*) balról és kicsit (*40px*) föntről nézzük a transzformációk hatását. A nézőpontot a 3-dimenziós teret adó, befoglaló *div* CSS-ébe bekódolva:

```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; -webkit-perspective:600px;
             perspective:600px; -webkit-perspective-origin:-
             50px -40px; perspective-origin:-50px -40px; }
#transzformalando { background-color:green; color:white; font-
           size:18px; font-family:"Courier New", Courier,
           monospace; font-weight:bold; text-align:center; }
```

Önmagában a perspektíva definiálása nem okoz a transzformálandó elemen semmilyen változást, és valamilyen térbeli transzformáció kódolásáig nincs is hatása, de a megadásával létrehoztuk a térbeli koordinátarendszert az eljövendő 3D-s műveletek számára.

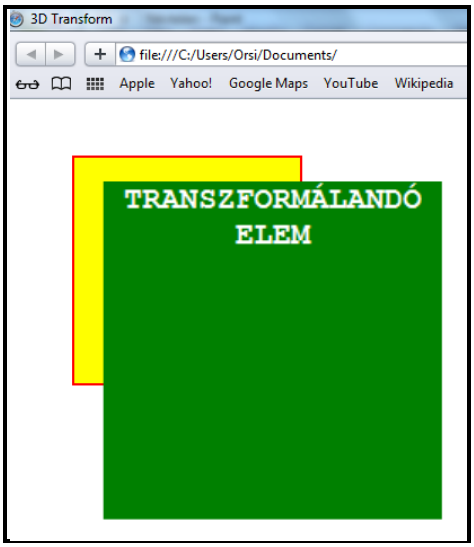
A síkbeli transzformációk *transform* (*x* és *y* tengelyekre értelmezett) tulajdonságainak a *z*-tengelyre való kiterjesztésével egyszerűen megvalósíthatók a térbeli transzformációk.

a) Az eltolás síkbeli megvalósítása és (pirossal) a térbeli kiterjesztése:

translateX(x)	vízszintes eltolás
translateY(y)	függőleges eltolás
translateZ(z)	a néző felé közelítés (pozitív érték) vagy távolítás (negatív érték); százalékos érték nem adható meg
translate(x, y)	síkbeli kétirányú eltolás összevont alakban
translate3d(x, y, z)	a térbeli eltolás összevont alakban (*x* és *y* értéke lehet %, *z* értéke nem lehet %)

Ha *200px*-el a néző felé toljuk a transzformálandó elemet, a kódolás és megjelenítés:

```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; -webkit-perspective:600px;
           perspective:600px; -webkit-perspective-origin:-50px
            -40px; perspective-origin:-50px -40px; }
#transzformalando { background-color:green; color:white; font-
             size:18px; font-family:"Courier New", Courier,
             monospace; font-weight:bold; text-align:center;
             -webkit-transform:translateZ(200px);
             transform:translateZ(200px); }
```



Bár a példa kódolásába az egyszerűbb érthetőség érdekében a részletezett alakot írtuk be, az összevont alak is használható, melynek a megfeleltetése:

translate3d(x, 0, 0) = translateX(x)
translate3d(0, y, 0) = translateY(y)
translate3d(0, 0, z) = translateZ(z) tehát jelen esetben *translate3d(0, 0, 200px);*
translate3d(x, y, 0) = translate(x, y)

Megjegyzések a fenti transzformációhoz:

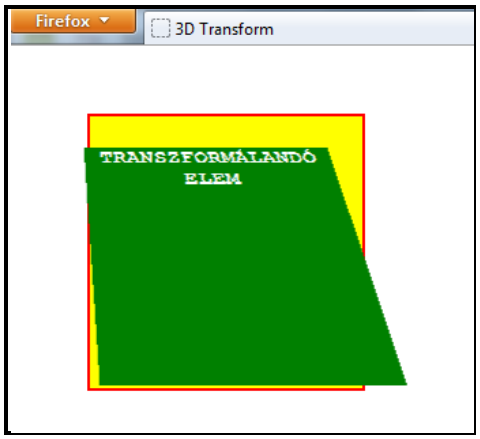
- ha a *perspective-origin*-t alapértelmezett (*0px 0px*) értékre állítjuk, a transzformált elem bal felső sarkának helyzete a művelet során nem változik
- ha *perspective* értékét növeljük, egyre kevésbé látszódik a *z*-irányú eltolás hatása, pl. *10000px*-nél már gyakorlatilag az eredeti helyzetben látszódik az elem
- ha a *perspective* értékét csökkentjük és/vagy a *translateZ* értékét növeljük (azaz csökken a nézőpont és az elem közötti *z*-érték), az elemen egyre erősebbnek látszódik a transzformáció hatása
- ha a *translateZ* értéke meghaladja a *perspective* értékét, eltűnik (mivel a néző mögé kerül) a transzformált elem
- ha negatív *z* eltolást kódolunk, az origo-hoz képest minden ellentétesen változik, mint pozitív értéknél

b) Az elforgatás síkbeli megvalósítása és (pirossal) a térbeli kiterjesztése:

rotate(deg) *x-y* tengelyek síkjában történő elforgatás
rotateX(deg) *x* tengely körüli elforgatás
rotateY(deg) *y* tengely körüli elforgatás
rotateZ(deg) *z*-tengely körüli elforgatás - megegyezik az *x-y* tengelyek síkjában történő elforgatással, azaz ***rotate(****deg****)***-el
rotate3d(x, y, z, deg) térbeli elforgatás összevont alakban

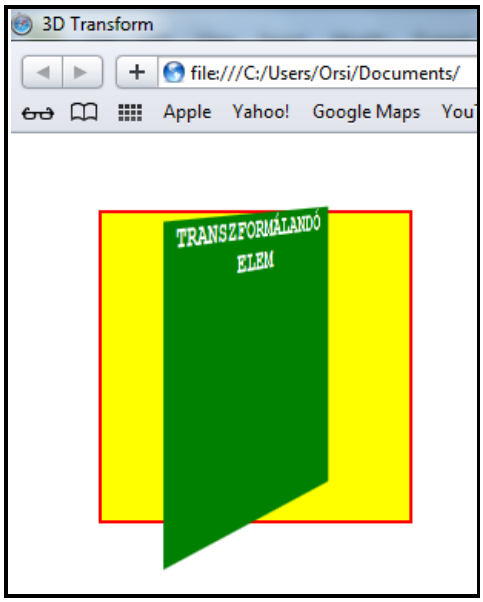
Az *x*-tengely mentén történő forgatás kódolása és megjelenítése:

```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; -webkit-perspective:600px;
             perspective:600px; -webkit-perspective-origin:
             -50px -40px; perspective-origin:-50px -40px; }
#transzformalando { background-color:green; color:white; font-
             size:18px; font-family:"Courier New", Courier,
             monospace; font-weight:bold; text-align:center;
             -webkit-transform:rotateX(45deg);
             transform:rotateX(45deg); }
```



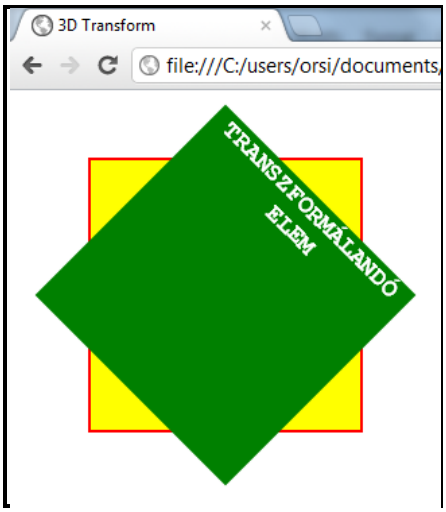
Az *y*-tengely mentén történő forgatás kódolása és megjelenítése:

```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; -webkit-perspective:600px;
             perspective:600px; -webkit-perspective-origin:
             -50px -40px; perspective-origin:-50px -40px; }
#transzformalando { background-color:green; color:white; font-
             size:18px; font-family:"Courier New", Courier,
             monospace; font-weight:bold; text-align:center;
             -webkit-transform:rotateY(45deg);
             transform:rotateY(45deg); }
```



Az *z*-tengely mentén történő forgatás kódolása és megjelenítése megegyezik a síkbeli transzformációknál kódolt *rotate(deg)* elforgatással:

```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; -webkit-perspective:600px;
           perspective:600px; -webkit-perspective-origin:-50px
            -40px; perspective-origin:-50px -40px; }
#transzformalando { background-color:green; color:white; font-
             size:18px; font-family:"Courier New", Courier,
             monospace; font-weight:bold; text-align:center;
             -webkit-transform:rotateZ(45deg);
             transform:rotateZ(45deg); }
```



c) Az átméretezés síkbeli megvalósítása és (pirossal) a térbeli kiterjesztése:

scaleX(x)	vízszintes irányú átméretezés
scaleY(y)	függőleges átméretezés
scaleZ(z)	a felhasználó irányába átméretezés
scale(x, y)	síkbeli kétirányú átméretezés összevont alakban
scale3d(x, y, z)	térbeli átméretezés összevont alakban

A térbeli összevont alak síkbeli megfeleltetése:

scale3d(x, 1, 1) = scaleX(x)
scale3d(1, y, 1) = scaleY(y)
scale3d(1, 1, z) = scaleZ(z)
scale3d(x, y, 1) = scale(x, y)

A *scaleZ* ill. *scale3d* nulla vastagságú elem z-irányú kiterjedésére vonatkozik, tehát nincsen valós tartalmuk, csak formai okokból kerültek bevezetésre. A *scale3d* mindhárom paraméterét meg kell adni, de a legutolsó tetszőleges érték lehet, hiszen úgyis nullával lesz szorozva.

d) Nincsen térbeli kiegészítésük az elferdítéseknek:

skewX(deg)	vízszintes elferdítés
skewY(deg)	függőleges elferdítés
skew(deg, deg)	síkbeli kétirányú elferdítés

e) A transzformációk origo-ja a síkbeli transzformációkhoz hasonlóan mindegyik műveletnél szintén szabadon elhelyezhető:

transform-origin(x, y)	síkbeli transzformáció origo-jának helyzete (alapértelmezett értéke (50% 50%) – ezt nem kell kódolni)
transform-origin(x, y, z)	térbeli transzformáció origo-jának helyzete (alapértelmezett értéke (50% 50% 0) – ezt nem kell kódolni)

A térbeli jellegből adódóan két további, a síkbeli műveleteknél nem értelmezett új tulajdonság egészíti ki a transzformációkat:

1) A hátlap láthatósága:

Mivel 3D-műveletekkel egy elem részben vagy teljes mértékben megfordítható, a hátoldala is láthatóvá válik. A **backface-visibility** tulajdonság alapértelmezetten láthatóan hagyja a hátoldalt (*visible*), ha el kívánjuk rejteni, a *hidden* értéket kell kódolni.

2) A transzformáció stílusa:

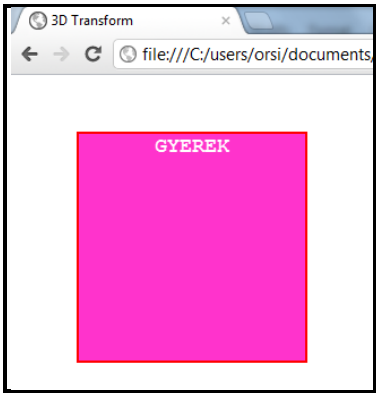
Amennyiben egy elemen térbeli transzformációt hajtunk végre, az elem gyerekeinek viselkedését a rajtuk végrehajtandó esetleges további transzformációk szempontjából két módon lehet definiálni.

Ezek bemutatásához egy újabb, (gyerek)*div*-et kell elhelyezni a transzformálandó elemben, így a HTML-kód:

```
<div id="befoglalo">
     <div id="transzformalando">
          <div id="gyerek">GYEREK</div>
     </div>
</div>
```

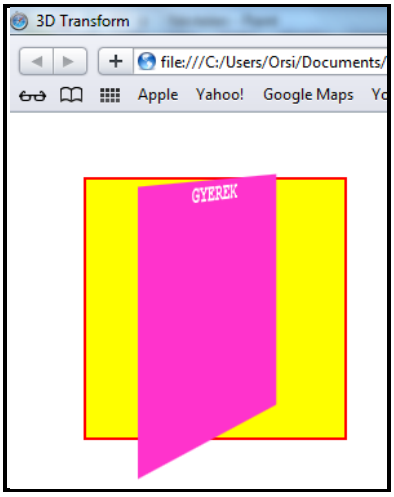
A CSS pedig (a rózsaszín gyerek-*div* mérete megegyezik a zöld szülő-*div* méretével és pontosan takarja azt – és mindketten pontosan takarják a sárga befoglaló *div*-et):

```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; -webkit-perspective:600px;
             -moz-perspective:600px; -webkit-perspective-
             origin:-50px -40px; -moz-perspective-origin:-50px
             -40px; }
#transzformalando { background-color:green; }
#gyerek { background-color:#F3C; color:white; font-size:18px;
          font-family:"Courier New", Courier, monospace; font-
          weight:bold; text-align:center; }
```



A szülő-*div*-et *45*°-al az *y*-tengely körül elforgatva:

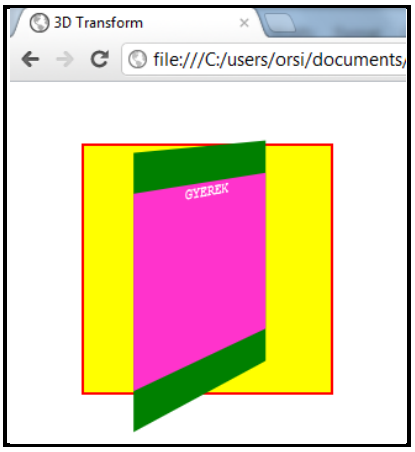
```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; -webkit-perspective:600px;
             perspective:600px; -webkit-perspective-origin:
             -50px -40px; perspective-origin:-50px -40px; }
#transzformalando { background-color:green; -webkit-transform:
             rotateY(45deg); transform:rotateY(45deg); }
#gyerek { background-color:#F3C; color:white; font-size:18px;
             font-family:"Courier New", Courier, monospace;
             font-weight:bold; text-align:center; }
```



A szülő a gyerekkel együtt transzformálódott, a gyerek továbbra is takarja a szülőt.

Most a gyereket forgatjuk *45*°-al az *x*-tengely körül:

```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; -webkit-perspective:600px;
             perspective:600px; -webkit-perspective-origin:-50px
             -40px; perspective-origin:-50px -40px; }
#transzformalando { background-color:green; -webkit-transform:
             rotateY(45deg); transform:rotateY(45deg); }
#gyerek { background-color:#F3C; color:white; font-size:18px;
             font-family:"Courier New", Courier, monospace; font-
             weight:bold; text-align:center; -webkit-transform:
                  rotateX(45deg); transform:rotateX(45deg); }
```

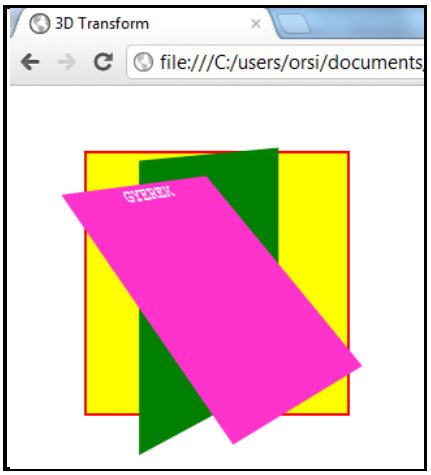


Az *x*-tengely mentén az elfordulás megtörtént, ezért kisebb a gyerek *div* magassága, de a szülő síkjában maradt laposan, térhatás nélkül, mert a gyereknek nem volt perspektivája (3-dimenziós koordinátarendszere). A *transform-style:flat*; ezt jelenti, és *flat* az alapértelmezett érték.

Közös 3-dimenziós térhez meg kell a szülőnek adni a *transform-style:preserve-3d* értéket, ekkor a gyerek is megőrzi a perspektívát, és térhatású lesz a gyerekre előírt transzformáció is:

```
div { height:200px; width:200px; }
#befoglalo { border:2px solid red; background-color:yellow;
             margin:50px; -webkit-perspective:600px;
            perspective:600px; -webkit-perspective-origin:-50px
            -40px; perspective-origin:-50px -40px; }
#transzformalando { background-color:green; -webkit-transform:
             rotateY(45deg); transform:rotateY(45deg); -webkit-
             transform-style:preserve-3d; transform-style:
             preserve-3d; }
#gyerek { background-color:#F3C; color:white; font-size:18px;
          font-family:"Courier New", Courier, monospace; font-
          weight:bold; text-align:center; -webkit-transform:
          rotateX(45deg); transform:rotateX(45deg); }
```

Megjegyzés: A fejezet elején már említett módon a *transform-style:preserve-3d* az a fontos tulajdonság/érték pár, amit az Internet Explorer-ek nem értelmeznek.



Lapos vagy térhatású elemek 3D-transzformációit folyamatos átmenetekkel vagy animációkkal kombinálva látványos effektek hozhatók létre. Erre egyszerű és jól használható, példa egy kép 180°-os elforgatása az *y*-tengelye körül *:hover* hatására *2mp*-es folyamatos átmenettel:

A HTML-kód:

```
<div id="befoglalo">
   <div id="kep">
      <div id="szine">
         <img src="IMG_1315.JPG" width="450px" height="300px">
      </div>
      <div id="fonakja">
          <p>A képpel kapcsolatos információk:</p>
          <p>.................................</p>
```

```
            <p>.................................</p>
            <p>.................................</p>
        </div>
    </div>
</div>
```

A „befoglaló” *div* biztosítja a 3-dimenziós teret, az abba beágyazott „kép” *div* tartalmazza a „színe” *div*-ben magát az *img* képet és a „fonákja” *div*-ben a hátoldali szöveget.

A CSS-kód:

```
#befoglalo { width:450px; height:300px; -webkit-perspective:
                1000px; perspective:1000px; }
#kep { width:100%; height:100%; -webkit-transform-style:
        preserve-3d; transform-style:preserve-3d; -webkit-
        transition:all 2s linear; transition:all 2s linear; }
#szine, #fonakja { position:absolute; width:100%; height:
        100%; -webkit-backface-visibility:hidden; backface-
        visibility:hidden; border:1px solid black; }
#fonakja { -webkit-transform:rotateY(180deg); transform:
        rotateY(180deg); padding:10px; color:white; text-
        align:center; background-color:grey; font-size:24px;
        font-weight:bold; }
#kep:hover { -webkit-transform:rotateY(180deg); transform:
        rotateY(180deg); }
```

Az egyes fenti kijelölésekhez tartozó meghatározások magyarázata:

- a befoglalóban megadtuk a kép méretét és a transzformációhoz a perspektívát. A perspektívára nincsen előírás, a kép méretéhez képest közepes távolságra és középre helyeztük (ezért nincsen a *perspective-origin* külön megadva).

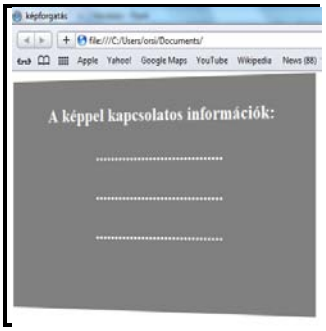
- a kép kitölti a befoglalóban már megadott helyét, a transzformáció stílusa *preserve-3d*, hiszen a színét és fonákját (a gyerekeit) majd meg akarjuk térben fordítani, és *2mp*-es egyenletes sebességű változást tervezünk előidézni. A folyamatos átmenet tulajdonságaként - a böngészőspecifikus előtagokba bonyolódás elkerülésére - az *all* (mindegyik) értéket egyszerűbb használni.

- a színe és fonákja pozíciója azért *absolute*, hogy egymáson legyenek (ne egymás után), a méretük a képméret maradjon, hátoldaluk ne látszódjon a transzformációt követően (hiszen a „fonákját” zavarná a „színe”). A keret nem szükségszerű, itt a kép világos széleit segít jobban elválasztani a háttértől.

- a fonákját elfordítjuk az *y*-tengelye körül 180°-al (a *transform-origin*-t nem kellett megadni), a rajta lévő szöveg formázása tetszőleges.

- a fenti 180°-os elforgatás a *:hover* hatására következik be.

Két pillanatkép a térbeli elfordulás során:



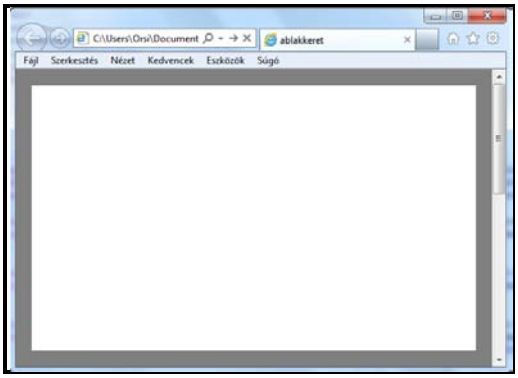
3.29. Speciális formázások pszeudo-elemekkel

A pszeudo-elemek egyik csoportját (*::before, ::after*) a forrásdokumentumban nem szereplő tartalmak beillesztésére lehet használni. Ebben a fejezetben beillesztendő tartalom nélküli (*content:””)*, tisztán formázási lehetőségeikre mutatunk be néhány hasznos példát.

1.) A weboldal kerettel díszítése, melynek helyzete független a kijelző eszköz méretétől és a tartalom görgetésétől

A weblap törzsére vonatkozik, tehát a *body*-ra definiált CSS-kódolással valósul meg:

```
body::after {
    content:"";
    position:fixed;
    top:0; left:0; right:0; bottom:0;
    border:20px solid grey; }
```



A *::before* vagy *::after* egyaránt használható, a szegély tulajdonságai a *border* értékeivel definiálhatók. Akár *border-radius* is megadható hozzá – ekkor ügyelni kell arra, hogy a lekerekített szegély és szögletes kijelzősarok közti területhez görgetéskor ne kerülhessen láthatóan mozgó háttér vagy tartalom.

Megjegyzés: Fenti keret négy db *div* HTML-kódolásával és a böngészőablak négy oldalához *position:fixed;* módon helyezésével is kialakítható. Annak előnye, hogy minden régi böngészővel is működik, hátránya, hogy csak a megjelenítés kedvéért hozunk létre HTML-kódot.

2.) Egy elem többszörös szegéllyel való ellátása

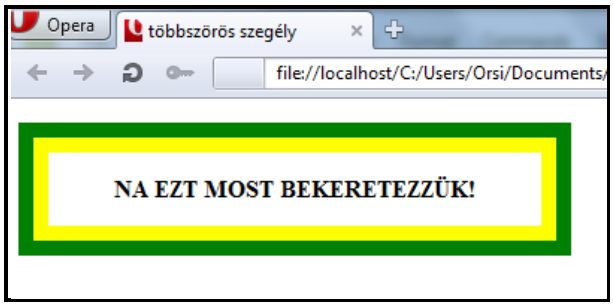
A *border* tulajdonsággal *egy* standard vonaltípusú szegély hozható létre. Pszeudo-elemmel *többszörös* szegély is kialakítható, melyből a dupla szegély kódolását mutatjuk be.

Legyen a bekeretezendő elem egy rövid paragrafus, azaz a HTML-kód:

```
<p>NA EZT MOST BEKERETEZZÜK!</p>
```

A CSS-kódolás:

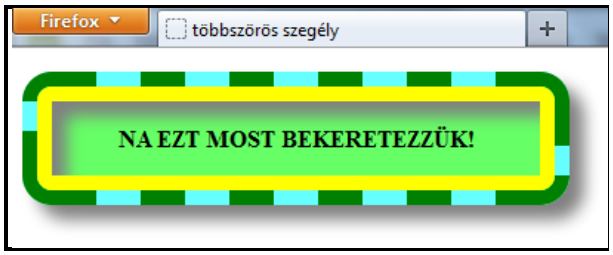
```
p { position:relative; padding:25px; border:10px solid green;
    width:300px; text-align:center; font-weight:bold; }
p::after { content:""; position:absolute; top:0; left:0;
right:0; bottom:0; border:10px solid yellow; }
```



A *::before* vagy *::after* egyaránt használható, a szegélyek tulajdonságai a *border* értékeivel definiálhatók.

A két szegély egymástól függetlenül is változtatható, különböző *border-radius* és *box-shadow* adható meg hozzájuk. Ha a második kijelöléshez külön adunk meg egy hátteret, *z-index* alkalmazása is szükséges, hogy a második háttér ne takarja el a tartalmat:

```
p { position:relative; padding:25px; border:10px dashed green;
    width:300px; text-align:center; font-weight:bold; border-
    radius:20px; box-shadow:10px 10px 10px grey; background-
    color:#6FF; z-index:1; }
p::after { content:""; position:absolute; top:0; left:0;
           right:0; bottom:0; border:10px solid yellow; border-
           radius:10px; box-shadow:10px 10px 10px grey inset;
           background-color:#6F6; z-index:-1; }
```



Megjegyzés: Dupla szegélyek egy újabb, a tartalmat beágyazó *div* HTML-kódolásával is kialakíthatók. Annak előnye, hogy minden régi böngészőben is működik, hátránya, hogy csak a megjelenítés kedvéért hozunk létre HTML-kódot.

3.) Sarok behajtása

Díszítő és térhatást keltő formázás a sarok visszahajtása („szamárfül”). Az alaphelyzetben a szöveget formázzuk, majd a *Standard vonaltípusú szegélyek* fejezetben leírtaknak megfelelően egy háromszöget alakítunk ki CSS-kódolással (azaz kép használata nélkül) a jobb felső saroknál.

Legyen tehát a HTML-kód:

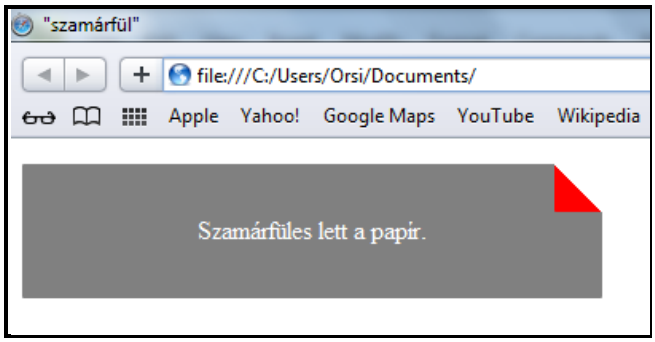
```
<p>Szamárfüles lett a papír.</p>
```

A CSS-kódolás pedig:

```
p { position:relative; width:300px; padding:2em; text-align:
    center; color:white; background-color:grey; }
```

```
p::before { content:""; position:absolute; top:0; right:0;
            border:15px solid transparent; background-color:red;
            border-color:white white transparent transparent; }
```

A kód *::before*-al vagy *::after*-al egyaránt működik, a második kijelölésben a visszahajtás nagyságát a *border* vastagságával, színét a *back-ground* színével lehet megadni. Ugyanitt a *border-color* első két értékének meg kell egyeznie a mindenkori háttér színével (jelen esetben ezért fehérek), különben a behajtott sarok helye fehéren marad.



A legkézenfekvőbb további formázás a sarkok lekerekítése, és a visszahajtott saroknál némi árnyékkal térhatás hozzáadása:

```
p { position:relative; width:300px; padding:2em; text-align:
    center; color:white; background-color:grey; border-radius:
    15px 0px 15px 15px; overflow:hidden; }
p::before { content:""; position:absolute; top:0; right:0;
            border:20px solid transparent; background-color:
            red; border-radius:0 0 0 15px; border-color:white
            white transparent transparent; box-shadow:0px 2px
          2px rgba(0,0,0,0.33), -4px 4px 4px rgba(0,0,0,0.33);
        }
```

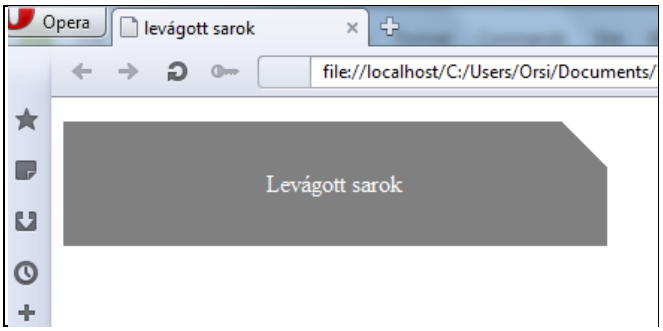
A *border-radius* értékek az első és második kijelölésben értelemszerűen azonosan változtak, a szamárfül méretén csak a látvány kedvéért változtattunk kicsit (*border:20px*…). A *box-shadow* hatására a behajtott sarok helyén is képződik egy függőleges árnyék, ezért kellett azt az *overflow:hidden*; bekódolásával elrejteni.

4.) Sarok levágása

A szögletes sarok visszahajtásának egyszerűsített esete, amikor nem kell a visszahajtás színével foglalkozni:

```
p { position:relative; width:300px; padding:2em; text-align:
    center; color:white; background-color:grey; }
```

```
p::before { content:""; position:absolute; top:0; right:0;
            border:15px solid transparent; border-color:white
            white transparent transparent; }
```



A *border* vastagságával lehet a levágás mértékét változtatni.

5.) Beszéd/gondolat buborék

A beszéd/gondolat buborék alaphelyzetben a szöveges tartalomból és a „hozzá vezető” - a *Standard vonaltípusú szegélyek* fejezetben leírtaknak megfelelően (azaz kép használata nélkül) kialakított - háromszögből áll.

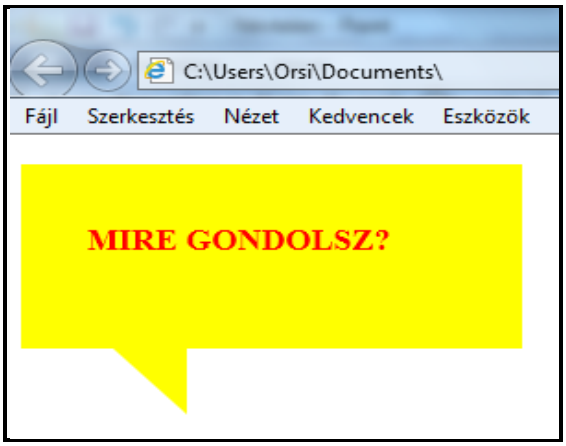
Legyen a szöveg/gondolat HTML-kódja:

```
<p>MIRE GONDOLSZ?</p>
```

A CSS-pedig:

```
p { position:relative; width:200px; height:40px; background-
    color:yellow; color:red; font-weight:bold; font-size:18px;
    padding:2em; }
p::before { content:""; position:absolute; bottom:-40px;
            left:50px; border-width:40px 0 0 40px; border-
            style:solid; border-color:yellow transparent; }
```

A formázott szöveghez/gondolathoz ismét egy kódolt háromszöget illesztettünk. A háromszög vízszintes helyzetét a *left* értékével, a nyílásszögét a *border-width* nem-nulla értékeivel, a magasságát a *bottom* értékével (ami legyen egyenlő a *border-width* első – *border-top-width* – értékével) lehet állítani. Mind *::before*, mind *::after* –el is működik.



A szöveget tartalmazó rész a szokványos módon kerekíthető, a háromszög pedig akár korongokkal helyettesíthető:

```
p { position:relative; width:200px; height:40px; padding:2em;
        color:red; background-color:yellow; font-weight:bold;
        font-size:18px; border-radius:50%; }
p::before { content:""; position:absolute; bottom:-30px;
          left:40px;width:30px; height:30px; background-
          color:yellow; border-radius:50%; }
p::after { content:""; position:absolute; bottom:-40px;
         left:20px; width:15px; height:15px; background-
         color:yellow; border-radius:50%; }
```

A korongok méretét és alakját a *width* és *height* értékei, helyzetüket a *bottom* és *left* értékei határozzák meg.

Ha van szegély, ügyelni kell, hogy ha átfedő részeket is kialakítunk, háttérrel takarásba kerüljenek az alsó szegélyek (a pszeudo-elemes kijelölésekhez hátteret kell kódolni):

```
p { position:relative; width:200px; height:40px; padding:2em;
         color:red; border:10px solid blue; font-weight:bold;
         font-size:18px; border-radius:50%; }
p::before { content:""; position:absolute; bottom:-40px;
        left:90px; width:45px; height:45px; border:10px solid
        blue; background-color:white; border-radius:50%; }
p::after { content:""; position:absolute; bottom:-70px;
        left:50px; width:25px; height:25px; border:10px solid
        blue; background-color:white; /* csak akkor kötelező,
       ha metszené a felette lévő kört */ border-radius:50%; }
```

Megjegyzés: Beszéd/gondolat buborékok nagy változatossággal alakíthatók ki, megvalósításuk *div*-ek bevezetésével is történhet. Annak viszont az a hátránya, hogy csak a megjelenítés kedvéért hozunk létre HTML-kódot.

6.) Szív

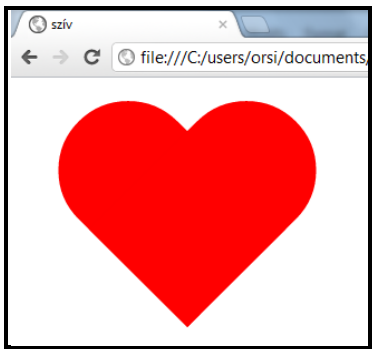
Egy szívhez szóló példát hagytunk a végére, melynek egy üres *div* a HTML-kódolása:

```
<div></div>
```

A CSS-kód (*Safari* és mobil böngészők miatt sajnos szükséges a –*webkit*-előtagos transzformáció kódolás is):

```
div { position:relative; }
div::before { position:absolute; content:""; left:50px; top:0;
    width:50px; height:80px; background-color:red; border-
    radius:50px 50px 0 0; -webkit-transform:rotate(-45deg);
    transform:rotate(-45deg); -webkit-transform-origin:0 100%;
    transform-origin:0 100%; }
div::after { position:absolute; content:""; left:0; top:0;
     width:50px; height:80px; background-color:red; border-
     radius:50px 50px 0 0; -webkit-transform:rotate(45deg);
     transform:rotate(45deg); -webkit-transform-origin:100%
     100%; transform-origin:100% 100%; }
```

Az eredeti *div*-nek (mely nem látszik, nincsen háttere vagy szegélye, méreteket sem kell adni neki) a pszeudo-elemekkel szabályos szívhez arányos méretet adtunk, a felső élét lekerekítettük, piros hátteret kapott, majd balra elforgattuk a bal alsó sarka körül, ill. jobbra a jobb alsó sarka körül 45°-45°-al és egymásra pozícionáltuk őket.



A méret változtatásánál a magasság/szélesség arányt érdemes megtartani, a *border-radius*-nak és a *div::before* kijelölésben a *left* értékének meg kell egyeznie a *width* szélességgel.

A *:hover* alkalmazásával a kurzorral a szív látványosan felnagyítható. De tekintettel arra, hogy a *:hover* az eredeti *div*-hez van rendelve, ekkor meg kell adni neki egy szélességet és magasságot a kurzoraktivitás részére – a szívet magában foglaló négyszög a *div*-nek ideiglenesen hátteret adva látható és beállítható, fenti méreteknél pl. *100px* x *80px*:

```
div:hover { -webkit-transform:scale(2.5, 2.5); transform:
            scale(2.5, 2.5); }
div { position:relative; width:100px; height:80px; }
```

Ha valamilyen tartalomba kerül a szív behelyezésre, szintén szükséges, hogy a *div* rendelkezzen méretekkel. Vegyük az alábbi példát:

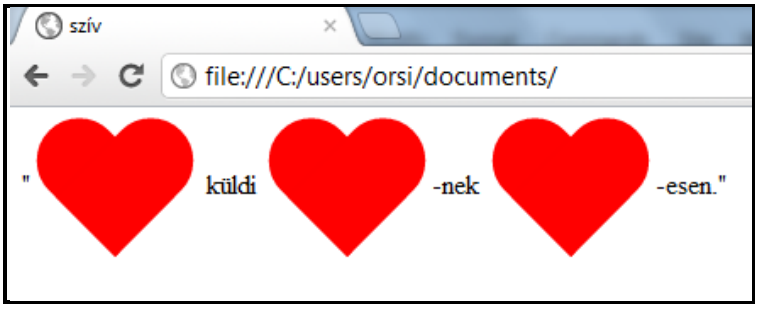
HTML-kód:

```
<q><div></div> küldi <div></div>-nek <div></div>-esen.</q>
```

CSS-kód:

```
div { position:relative; width:100px; height:80px;
      display:inline-block; vertical-align:middle; }
```

A többi (*div::before*, *div::after*, opcionálisan a *div:hover*-es) kódok változatlanok maradnak.



Szívdobogás a *scale* átméretezés animálásával hozható létre (*Chrome*, *Opera* és *Safari* részére –*webkit*- előtagos kódolás is szükséges):

```
@-webkit-keyframes dobog {
     from { -webkit-transform:none; }
      50% { -webkit-transform:scale(1.2); }
       to { -webkit-transform:none; }
}
div { -webkit-animation:dobog 1s infinite; }
@keyframes dobog {
     from { transform:none; }
      50% { transform:scale(1.2); }
       to { transform:none; }
}
div { animation:dobog 1s infinite; }
```

Megjegyzés: A tárgyalt két pszeudo-elemet az IE8 csak ***egy*** kettősponttal ismeri fel. Tekintettel arra, hogy az összes többi korszerű böngészővel mind *egy*, mind *két* kettősponttal egyaránt használhatók, az egyik lehetséges választás a régi szabvány szerinti alkalmazás az IE8 kedvéért. Ennek akkor lehet létjogosultsága, ha az IE8 által is helyesen értelmezett (vagyis csak CSS 2.1) tulajdonságokat rendelünk hozzájuk. Tekintettel az Internet Explorer 8 egyéb korlátjaira és várható marginalizálódására, középtávon célszerű a szabványos két kettőspont alkalmazásának megtanulása és alkalmazása.

3.30. CSS alapbeállítás

Az egyes böngészők alapértelmezetten számos tulajdonságot egymáshoz képest eltérő értékkel jelenít(h)e(t)nek meg. A weblap egységes megjelenése érdekében a CSS alapbeállítással (*CSS Reset*) – az alapbeállítás hatókörén belül – a böngészők hatásainak eltérései kiküszöbölhetők, ill. az értékek egységesíthetők.

Mindenre kiterjedő, egyedül üdvözítő alapbeállítás nincsen – itt négy megoldás kerül ismertetésre, amelyek egy-az-egyben, vagy a felhasználói igényeknek megfelelően módosítva –előnyeik és hátrányaik ismeretében –felhasználhatók, ill. ötlet meríthető belőlük:

a) Alapbeállítás univerzális kijelölővel (*Universal Selector CSS Reset*):

```
* {
   margin:0;
   padding:0;
   border:0;
   outline:0;
   font-size:100%;
   line-height:1.5em;
   text-decoration:none;
   vertical-align:baseline; }
```

Legegyszerűbb esetben csak a *margin*-ra és *padding*-re, majdnem ugyanilyen gyakran a *border*-el és *outline*-al kiegészítve adják meg, és további tulajdonságok/értékek definiálásával az alapbeállítás egyszerűen bővíthető.

Előnye, hogy biztosan nem felejtünk ki semmit, hiszen egy adott tulajdonság/érték pár minden elemre vonatkozik– még az ezután kitalálandó/bevezetésre kerülő HTML-címkékre is. Minimális a kódolási munka és az alapbeállítás fájlmérete.

Hátránya viszont, hogy minden egyes elemre mindig meg kell adni a lenullázott értékek formázását, nem működik az öröklődés (a szülő-gyerek viszonyt felülírja a *reset*), az űrlapok *input* elemei és *gombjai* szétesnek (mindent nekünk kell megformázni).

b) Meyer-féle alapbeállítás:

```
html, body, div, span, object, iframe, h1, h2, h3, h4, h5, h6,
p, blockquote, pre, a, abbr, address, cite, code, del, dfn,
em, img, ins, kbd, q, s, samp, small, strong, sub, sup, var,
b, u, i, dl, dt, dd, ol, ul, li, fieldset, form, label,
legend, table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, embed, figure, figcaption, footer, header,
hgroup, nav, section, time, mark, audio, video {
            border:0; font-size:100%; font:inherit;
            vertical-align:baseline; margin:0; padding:0; }
article, aside, figcaption, figure, footer, header, hgroup,
nav, section {
            display:block; }
body { line-height:1; }
ol, ul { list-style:none; }
blockquote, q { quotes:none; }
blockquote::before, blockquote::after, q::before, q::after {
            content:none; }
table { border-collapse:collapse; border-spacing:0; }
```

A HTML-címkék szinte mindegyikére ad valamilyen (általában az alapértelmezettel egyező) alapbeállítást, de sok tulajdonság alapértéke természetesen definiálatlan marad.

c) HTML5 Doctor CSS Reset:

```
html, body, div, span, object, iframe, h1, h2, h3, h4, h5, h6,
p, blockquote, pre, abbr, address, cite, code, del, dfn, em,
img, ins, kbd, q, samp, small, strong, sub, sup, var, b, i,
dl, dt, dd, ol, ul, li, fieldset, form, label, legend, table,
caption, tbody, tfoot, thead, tr, th, td, article, aside,
figcaption, figure, footer, header, hgroup, nav, section,
time, mark, audio, video {
     margin:0; padding:0; border:0; outline:0; font-size:100%;
     vertical-align:baseline; back-ground:transparent; }
body { line-height:1; }
article, aside, figcaption, figure, footer, header, hgroup,
nav, section { display:block; }
nav ul { list-style:none; }
blockquote, q { quotes:none; }
blockquote::before, blockquote::after, q::before, q::after {
              content:none; }
a { margin:0; padding:0; font-size:100%; vertical-align:
    baseline; background:transparent; }
ins { background-color:#ff9; color:#000; text-decoration:none; }
mark { background-color:#ff9; color:#000; font-style:italic;
      font-weight:bold; }
del { text-decoration:line-through; }
abbr[title], dfn[title] {
             border-bottom:1px dotted; cursor:help; }
table { border-collapse:collapse; border-spacing:0; }
input, select { vertical-align:middle; }
hr { display:block; height:1px; border:0; border-top:1px solid
     #ccc; margin:1em 0; padding:0; }
```

A Meyer-féle alapbeállítás kiegészítésének tekinthető, melyben:

- az elemekhez általában átlátszó hátteret ad meg
- az *ins*-nek és *mark*-nak saját hátteret definiál
- az *ins*-nek, *del*-nek és *mark*-nak saját betűtulajdonságokat ír elő
- a vízszintes elválasztó vonal formázását rögzíti
- érintetlenül hagyja a lista-címkék felsorolásjeleit, csak a *nav ul* navigációs alkalmazásnál törli a felsorolásjelet
- az *abbr* és *dfn* elemeket szaggatott szegéllyel és kérdőjel alakú kurzorral látja el, ha van *title* jellemzőjük, mely magyarázatot ad az elem jelentéséhez
- az űrlap beviteli elemeket függőlegesen középre igazítja

d) A **leggyakrabban alkalmazott**, bonyodalmat nem okozó, minimális komplexitású és fájlméretű CSS alapbeállítás a böngészők alapértelmezetten eltérő margóbeállításainak kiküszöbölését szolgáló alábbi kódolás:

```
body, p, h1, h2, h3, table, img, ol, ul, form { margin:0;
                                                padding:0; }
```

3.31. Űrlap formázása

Az űrlap szerkezete és az egyes elemek fő kialakítása kötött, csakúgy mint a naptár vagy a csúszka megjelenítése, de a betűk, szegélyek, belső és külső margók, színek, háttérszínek és háttérképek, szélességek és pozíciók a CSS részben ismertetett módszerekkel formázhatók.

A HTML-rész *Űrlapok* fejezetében kódolt és alapértelmezetten bemutatott pizzarendelős űrlapon néhány formázási művelet végrehajtásával demonstráljuk a megjelenítés kialakításának lehetőségeit:

- CSS alapbeállítás és a törzs 500px szélesen középre pozícionálva:

```
body, form, p, h1, { margin:0; padding:0; }
body { width:500px; margin-right:auto; margin-left:auto; }
```

- az űrlap kitölti a törzset, fekete lekerekített szegéllyel van bekeretezve, és sugárirányú színátmenetes hátteret kap:

```
form { width:100%; border:5px solid black; border-radius:25px;
       padding:30px; background-image:radial-gradient(yellow,
       maroon); }
```

- az űrlapelem-csoportok lazább helyzetbe állítása és a keretük formázása:

```
fieldset { margin-right:8%; margin-left:8%; margin-top:5%;
           margin-bottom:5%; border:3px solid aqua; border-
           radius:25px; }
```

- az űrlap címének formázása:

```
h1 { font-family:Georgia, serif; font-weight:bold; font-
     stretch:expanded; font-style:oblique; font-size:40px;
     color:silver; text-align:center; }
```

- a beviteli mezők kiemelése színes kerettel , amennyiben a kurzorral fókuszba helyezi a felhasználó:

```
input:focus, textarea:focus { border:2px solid blue; }
```

- visszaállító és elküldő gombok helyzetének beállítása és formázásuk:

```
input[type="reset"] {
    width:150px; height:30px; background-image:linear-gradient
    (#060, #0f0, #060); border-radius:10px; font-weight:bold;
    color:yellow; margin-left:40px; margin-top:15px; }
input[type="submit"] {
    width:150px; height:30px; background-image:linear-gradient
    (#06f, #0ff, #06f); border-radius:10px; font-weight:bold;
    color:yellow; margin-left:120px; margin-top:15px; }
```

- elemfeliratok és elemcsoport-címek formázása:

```
label { font-size:18px; color:green; font-weight:bold; }
legend { font-size:25px; color:blue; }
```

- jelölőnégyzetek és választógombok áthelyezése:

```
input[type="radio"], input[type="checkbox"] { margin-left:
                                                150px; }
```

- a többsoros szövegmező méretének és helyzetének változtatása:

```
textarea { margin-left:3px; width:250px; vertical-align:
                                                middle; }
```

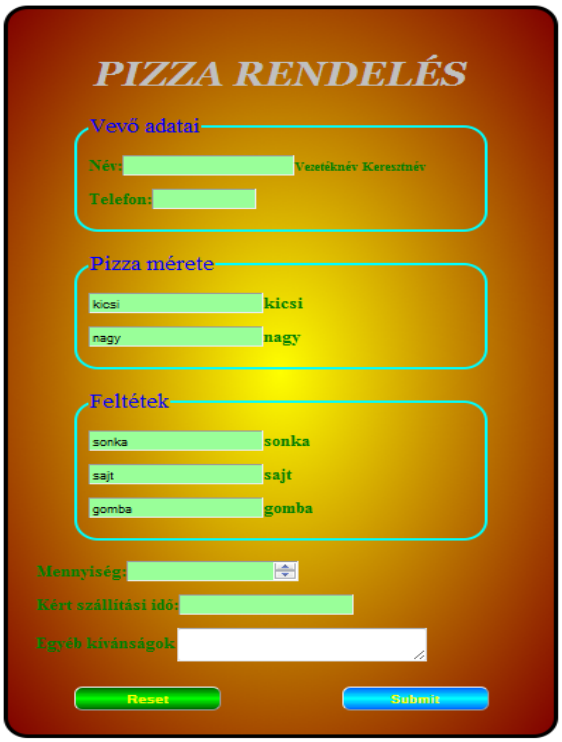
- a beviteli mezők hátterének beállítása:

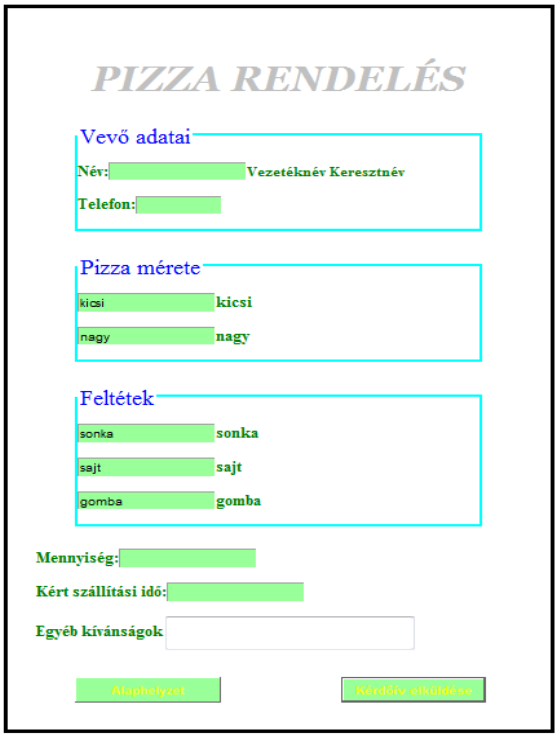
```
input { background-color:#9f9; }
```

- az Internet Explorer-ekben a szövegmező görgetősávja – a többi böngészőhöz hasonlóan – csak akkor jelenjen meg, ha a szöveg túlcsordul a szövegmezőn:

```
textarea { overflow:auto; }
```

A formázás természetesen még tovább folytatható – az eddigiek alapján az űrlap ilyenné változott (bal oldali kép a valamennyi tulajdonságot értelmező böngészőben, jobbra az Internet Explorer 8-ban):





A felhasznált CSS3 tulajdonságokat nem (Internet Explorer 8) vagy csak részben értelmező (Internet Explorer 9) böngészők tartalmilag szintén helyesen, de a formázást tekintve kevésbé választékosan jelenítik meg az űrlapot.

Megjegyzés: Ilyen esetben egy fix háttérszín adása a színátmenet kódja ***előtt*** (melyet értelmezni fognak a régebbi böngészők, de figyelmen kívül hagynak az újak) javít a megjelenítésen.

3.32. Legördülő menük

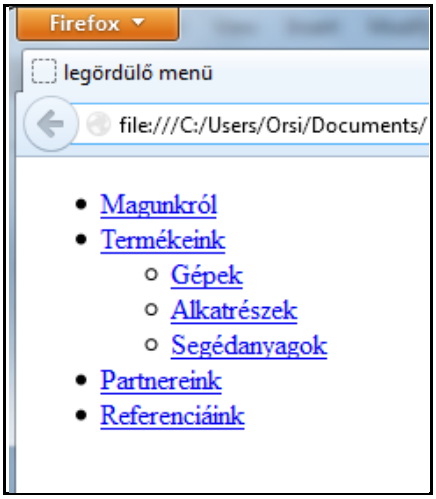
A le- vagy kigördülő menüket hagyományosan szkript-nyelvekkel állítják elő. Mindez CSS-el is kódolható, és a forgalomban lévő böngészők gond nélkül megjelenítik az ilyen módon kialakított navigációs (hivatkozási) menüsorokat vagy oszlopokat.

HTML-ben számozatlan/felsorolási listaként kódolunk hivatkozásokat, melyből CSS-el navigációs sort vagy oszlopot alakítunk ki. Példánkban egy weboldal navigációs sávjában elhelyezett menühöz hozunk létre legördülő navigációs oszlopot, a weblap tartalmi részével és a hivatkozási végpontokkal az áttekinthetőség kedvéért nem foglalkozunk.

A navigációs sávban elhelyezett menü - melyben beágyazott listaként szerepel a majdan legördülő menüoszlop - HTML-kódja (stílus még nincsen megadva):

```
<!doctype html>
<html lang="hu">
  <head>
    <meta charset="utf-8">
    <title>legördülő menü</title>
    <style></style>
  </head>
  <body>
    <nav>
      <ul class="menu">
        <li><a href=".........."">Magunkról</a></li>
        <li><a href=".........."">Termékeink</a>
          <ul>
            <li><a href=".........."">Gépek</a></li>
            <li><a href=".........."">Alkatrészek</a></li>
            <li><a href=".........."">Segédanyagok</a></li>
          </ul></li>
        <li><a href=".........."">Partnereink</a></li>
        <li><a href=".........."">Referenciáink</a></li>
      </ul>
    </nav>
  </body>
  </html>
```

A stílus nélküli HTML-kód megjelenítése tehát:



A fenti kód HTML-részéhez a továbbiakban nem nyúlunk, csak a <*style*> stílust alakítgatjuk.

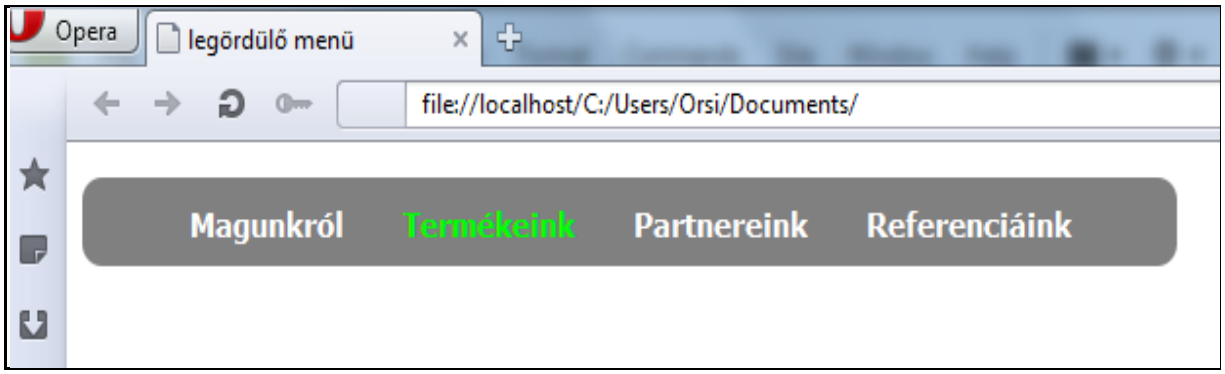
Első lépésben a legördülő rész nélküli menüsort alakítjuk ki, ezért egyelőre a *display* tulajdonság ideiglenes beiktatásával (***.menu ul { display:none; }***) láthatatlanná tesszük a beágyazott listát. Az „eltüntetés” és a többi rész formázása:

```
<style>
      .menu ul { display:none; }
      .menu { height:40px; width:520px; background:grey;
              border-radius:10px; }
      .menu li { position:relative; list-style:none;
              float:left; height:40px; }
      .menu li a { display:block; padding:0 15px; margin:6px 0;
              line-height:30px; text-decoration:none; font-
              family:Tahoma, Geneva, sans-serif; font-weight:
              bold; font-size:15px; color:white; }
      .menu li:hover > a { color:lime; }
</style>
```

A fenti meghatározások megadják:

- a menüsor méretét, hátterét, és lekerekítik a sarkokat
- a listaelemek helyzetét és eltünteti a felsorolásjeleket
- formázzák a hivatkozások megjelenítését
- az elemek fölé kerülő kurzor hatására színváltozást hoznak létre.

A legördülő rész nélküli navigációs sor megjelenítése (a kurzor a „Termékek” felett van, de mindegyik elem felett ugyanezt a színváltozást váltja ki):



A második lépésben formázzuk a legördülő részt, vagyis:

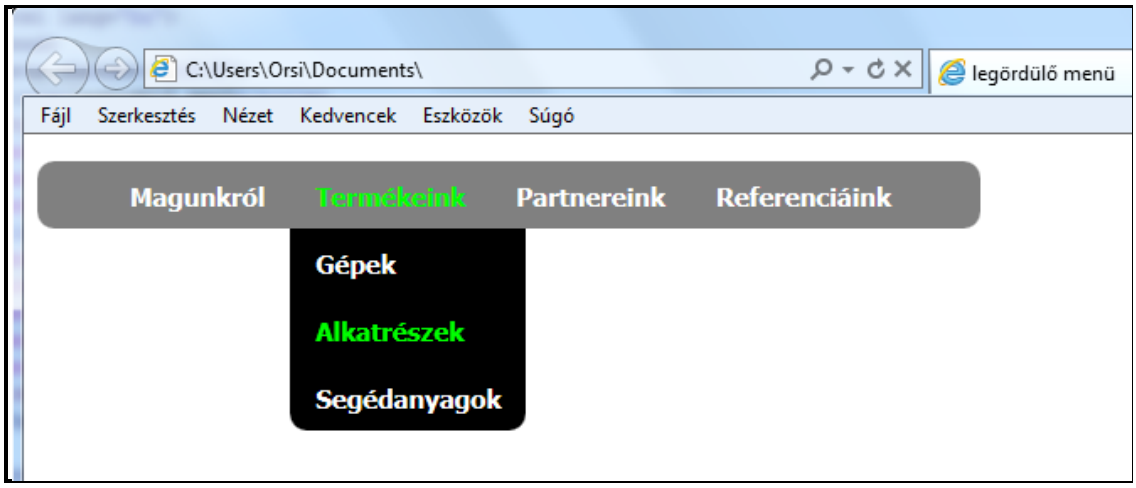
- „visszatesszük” a beágyazott listát (a pirossal jelölt ***.menu ul { display:none; }*** kód-sort törölni kell)
- formázzuk a beágyazott listát is, beleértve a kurzor színváltó hatását is
- a beágyazott listát a vonatkozó menüpont alá pozícionáljuk (ezért abszolút a helyzete, és a pozíciója egybeesik a „Termékek” alsó élével)
- a láthatóságát az átlátszóság (*opacity*) révén a kurzor helyzetétől tesszük függővé

A második lépés kódjait (és a kivett *display:none* –t) pirossal jelölve a teljes CSS stílus:

```
<style>
      .menu ul { display:none; }          /* ezt kivenni! */
      .menu { height:40px; width:520px; background:grey;
              border-radius:10px; }
```

```
    .menu li { position:relative; list-style:none;
               float:left; height:40px; }
    .menu li a { display:block; padding:0 15px; margin:6px 0;
               line-height:30px; text-decoration:none; font-
               family:Tahoma, Geneva, sans-serif; font-
               weight:bold; font-size:15px; color:white; }
    .menu li:hover > a { color:lime; }
    .menu ul { position:absolute; padding:0; top:40px;
              left:0; background:black; border-radius:0 0
              10px 10px; opacity:0; }
    .menu li:hover > ul { opacity:1; }
    .menu ul li { height:0; overflow:hidden; padding:0; }
    .menu li:hover > ul li { height:40px; overflow:visible; }
</style>
```

A legördülő menüben – a színből is látható – a kurzor éppen az „Alkatrészek” felett van (mely újabb lista-beágyazások révén további menü-ágakkal bővíthető):



3.33. Weblap elrendezések

A weblapok tartalmát különböző kimeneti eszközökön (leggyakrabban valamilyen kijelzőn vagy nyomtatón) jelenítik meg. Egy adott weblap használatának „élvezeti értékét” a weblap elrendezése/formázása és a megjelenítő eszköz összhangja határozza meg. Tekintettel arra, hogy az esetek többségében a felhasználók eltérő méretű, felbontású és színgazdagságú eszközöket használnak, a weblap használhatóságát jelentősen növeli az ugyanahhoz a tartalomhoz rendelt, de a tipikus eszközökhöz jól alkalmazkodó, egymástól eltérő stílusok meghatározása.

A képernyőn való megjelenítés szempontjából az az előnyös, ha a weboldal szélessége közel azonos a mindenkori kijelző képernyő felbontásával. Ha kisebb a képernyő felbontása, csak gördítősávval tekinthető meg a tartalom, ha nagyobb a felbontás, akkor kihasználatlan, üres felület is keletkezik a képernyőn.

Az elrendezési kialakítás öt fő technikája:

a) rögzített elrendezés

A weboldal szélessége abszolút értékben (általában *px*-ben) van megadva, szélessége a felhasználó képernyőfelbontásától független. Automatikus görgetősáv illeszti a megadottnál kisebb felbontáshoz, görgetéskor a megjelenítés arányai nem torzulnak.

A magyar internet-használatban jelenleg alkalmazott képernyőfelbontások körülbelüli megoszlása (csak az első tíz van felsorolva, melyek kb. 84%-os részarányt adnak ki):

1024 x 768	1366 x 768	1280 x1024	1280 x 800	1440 x 900	1920 x1080	1680 x1050	1152 x 864	1024 x 600	1600 x 900
21%	19%	12%	12%	6%	5%	3%	2%	2%	2%

Fentiek alapján a weboldalt 1.100-1.300 *px* szélesre célszerű tervezni.

b) rugalmas vagy folyékony elrendezés

A rugalmas elrendezés a folyékony (*liquid*) elrendezés és nyújtható (*elastic*) elrendezés gyűjtőneve. Tekintettel arra, hogy a nyújtható elrendezés a gyakorlatban alig használatos, a rugalmas a folyékony elrendezés szinonimájaként használható.

A rugalmas elrendezésben a weboldal szélességének nincsen képpontban meghatározott értéke, hanem a felhasználó képernyője felbontásának százalékában (nyújtható elrendezéskor *em*-ben, azaz betűméretben) van megadva. A weboldal szélességét 100%-ra kódolva a rugalmas elrendezés minden kijelzőn illeszkedik a rendelkezésre álló vízszintes mérethez, és - amennyiben a különböző belső *div*-ek is százalékosan vannak kódolva – a tartalom minden részének szélessége arányosan változik az össz-szélességgel.

A fenti előnyök mellett viszont kisebb felbontások felé haladva a tartalom egyre sűrűbbé válik, a fontos és kevésbé fontos részek differenciálatlanul keskenyednek, esetleg a fontos szöveg már olvashatatlanná válik, míg egy kevésbé fontos kép még szükségtelenül jól látható marad, stb. A rugalmas elrendezés tehát egy kb. 20%-os felbontástartomány változásban biztosíthatja a képernyő és weboldal-elrendezés összhangját, de az egyes tartalomrészeknek a kijelzőeszköz szerinti megjelenítésbeli megkülönböztetése már további finomítást ígényel.

c) hibrid vagy kevert elrendezés

A hibrid elrendezés a rugalmas és rögzített elrendezés kevert alkalmazása egy weboldalon belül. Pl. a fontosabb, vagy méretváltozásra érzékenyebb rész(eke)t rögzített szélességgel, a kevésbe fontosnak tartottakat rugalmas szélességgel kódoljuk, kompenzálva a tisztán rugalmas elrendezésnél jelzett differenciálatlanságot.

Egy egyszerű hibrid elrendezésre példa az alábbi, fejlécből, láblécből, navigációs sávból és egy (*article* és *aside* –ra) kettéosztott tartalmi részből álló weblap. Az *article* a fontos, széles felbontási határok között változatlan szélességű tartalomrész, az *aside*-nak kell (egy minimálisan megadott értékig) igazodnia a rendelkezésre álló vízszintes kiterjedéshez. A látszólag hosszú kódban a *lipsum*-ra azért van szükség, hogy legyen valamilyen tartalma a weblapnak – enélkül nem lenne függőleges kiterjedése.

A *main* burkoló konténer tartja egyben az elrendezést. A fejléc, lábléc és navigációs sáv tulajdonság/értékeit két külön helyen adtam meg (ez nem szükséges, de így próbáltam demonstrálni, mi tartozik a szerkezethez, és mi vonatkozik a – zölddel írt – alkalmi szöveg formázásához). A tartalmat hordozó *article* és *aside* tárolóelemeket a *wrapper* tartja egyben. A *body* szélességének egy 800 *px*-es alsó határt, az *article*-nek 600 *px* –es rögzített szélességet szabtam (így az *aside* sem tűnhet teljesen el a kijelző vízszintes méretének további csökkenésekor):

```
<!doctype html>
<html lang="hu">
   <head>
      <meta charset=8859-2>
      <title>vizszint. fix-rugalmas</title>
      <style>
         body, p, h1, h2, h3, table, img, ol, ul, form { margin:0; padding:0; }
         body { min-width:800px; }
            #main { width:100%; clear:left; }
               header { clear:left; width:100%; height:40px; background:blue; }
               nav { clear:left; width:100%; height:40px; background:red; }
               #wrapper { float:left; width:100%; }
                 aside { margin-left:600px; background:yellow; }
                 article { float: left; width: 600px; margin-left: -100%; background:green; }
               footer { clear:left; width:100%; height:40px; background:black; }
              header { padding:10px; text-align:center; color:white; }
              nav{ padding:10px; text-align:center; color:white; }
              footer { color:white; text-align:center; padding:10px; }
      </style>
   </head>
   <body>
      <div id="main">
         <header>HEADER = FEJLÉC</div></header>
         <nav>NAV = NAVIGÁCIÓS SÁV</nav>
         <div id="wrapper">
            <aside><h1>"aside" oszlop: rugalmas</h1><br><p>Proin sed elit felis, in facilisis arcu. Aliquam malesuada, magna quis eleifend pharetra, nunc nunc pretium sapien, mattis consequat nulla elit a nulla. Nullam tincidunt mattis odio. Morbi dapibus sem sit amet nibh lobortis sed interdum est lacinia. Morbi inconvallis lectus. Ut ac hendrerit neque. Nullam pellentesque, arcu non mattis egestas, arcu tortor convallis orci, ac ultricies urna lacus at nisi. Curabitur dignissim nunc quis libero convallis feugiat. Mauris consequat, lorem a feugiat aliquet, nibh mauris lobortis risus, in porta felis augue attortor. Vivamus ullamcorper interdum mattis.</p></aside></div>
            <article><h1>"article" oszlop: fix 600px</h1><p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed tempus augue in erat dignissim accumsan. Suspendisse eu
```

*vestibulum dui. Phasellus tortor mauris, accumsan nec dapibus a, sodales nec ante. Duis sollicitudin tortor id dui convallis vel sagittis augue ornare. Aenean quam urna, tempor in ultrices id, interdum et sem. Aliquam sollicitudin faucibus magna ut porta. Etiam feugiat venenatis mauris, sed posuere erat mollis eu. Ut sit amet leo nisi, quis aliquam felis. Sed sit amet urna vel urna tincidunt elementum eget vitae dolor. Nunc vitae imperdiet augue. Proin ultrices libero vitae enim lacinia consequat.</**p**>*

*<**p**>Vivamus eu lorem non tortor euismod laoreet in ut purus. Morbi laoreet mollis enim, in bibendum eros consectetur nec. Vestibulum ultricies adipiscing lectus, vel condimentum neque facilisis non. In hac habitasse platea dictumst. In pulvinar augue ac nisi placerat et varius arcu imperdiet. Nulla facilisi. Aenean quis odio erat, nec accumsan lectus. Curabitur gravida, felis ut vestibulum adipiscing, dui elit rutrum nulla, nec interdum ipsum enim id tortor. Sed odio nisi, sodales vel cursus a, porta eu ante. Fusce nec arcu rhoncus eros commodo fringilla. Suspendisse nec pellentesque turpis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Vestibulum a arcu eu arcu commodo dapibus scelerisque ac ligula. Duis neque lacus, mollis a faucibus eget, consectetur eget ligula.</**p**>*

*<**p**>Quisque consectetur, mauris a iaculis eleifend, metus dolor consequat enim, eu consec-tetur ante lorem sed ligula. Vivamus in suscipit leo. Nullam consectetur vulputate justo nec viverra. Sed porttitor nulla ac ligula molestie euismod. Proin laoreet ullamcorper tempus. </**p**></**article**></**div**>*

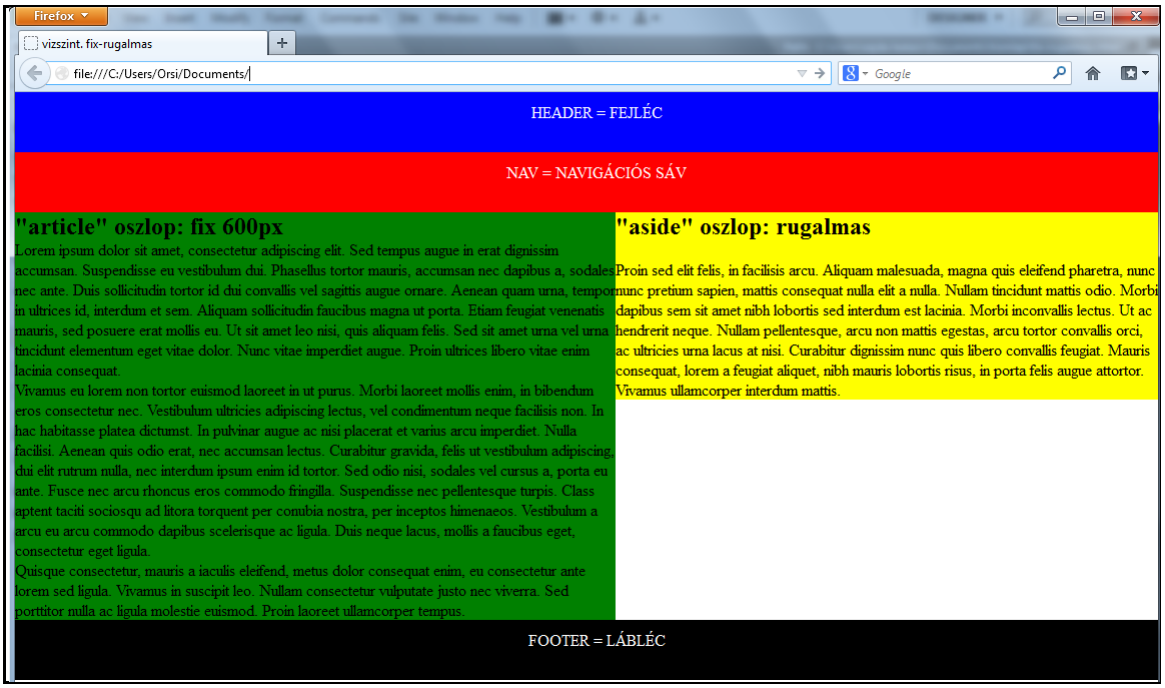
*<**footer**>FOOTER = LÁBLÉC</**footer**>*

*</**div**>*

*</**body**>*

*</**html**>*

A rugalmas elrendezéshez képest a fontos *article* részt az *aside* rovására – bizonyos képernyőfelbontás tartományban - változatlan formában meg lehetett őrizni. A kódolás megjelenítése az alábbi képen látható - a böngésző keretének változtatásával az elrendezés alkalmazkodása megfigyelhető:



d) a *media* jellemző alkalmazása

A *media* jellemző a kimeneti (megjelenítő) eszközre vonatkozó feltétel(ek) leírására használatos. Két logikai állapot közül választ – vagy egyezik a megjelenítő eszköz a megadottal, és akkor érvényes a hozzárendelt stílus, vagy nem egyezik, és akkor az a stílus nem érvényesül. A *media* jellemzőben felsorolásként egyszerre több média paraméter is definiálható, a felsorolásban a vessző a „logikai vagy”-ot (tehát a több feltétel közül legalább egynek a teljesülését), az *and* a „logikai és”-t (tehát a több feltétel egyidejű teljesülését) jelenti.

A képernyős megjelenítést a *screen* érték definiálja, melynek további paraméterei pontosíthatók, mint pl. *width, height, device-width, device-height, orientation* (portrait, landscape = álló, fekvő), *aspect-ratio* (4/3, 3/2), *device-aspect-ratio* (16/9, 1280/720), *color* (színenkénti bitek száma), *monochrome* (pixelenkénti bitek száma).

A képernyő tulajdonságok megadhatók HTML-ként (a *link* címkével külső CSS stíluslap csatolásaként – lásd a HTML Összefoglaló végét) vagy CSS kódként is (*@media screen* …….).

Példa egy 1100 *px* és 1300 *px* közötti szélességű képernyő esetén érvényes, a *@media* utasításban megadott megjelenítési stílusnak a definiálására:

@media *screen and (min-width:1100px) and (max-width:1300px)*

{ ……..…

…..…..…..…..…..

…….. **}**

Amennyiben a *media* –val 3-4 stíluslapot rendelünk egy dokumentumhoz, a szinte teljes, használatban lévő képernyő skála lefedhető nem túlfeszítetten alkalmazkodó elrendezésekkel.

Megjegyzés: A *media* jellemző *print* értékével a *d)* pontban leírtakkal analóg módon a képernyőstől eltérő, kimondottan nyomtatásra optimalizált elrendezés is csatolható a dokumentumhoz. Néhány fontos szempont a nyomtatáshoz készített stílusok kialakításához:

- a görgethető képernyőképpel (*continuous media*) ellentétben a nyomtatott képnek (*paged media*) adott papírmérethez kell alkalmazkodnia
- a nyomtatáskor a háttérképek és háttérszínek általában csak felesleges plusz költséget jelentenek
- a mozgó képek értelmezhetetlenek, a linkek általában feleslegesek
- a betűméretek és betűtípusok olvashatósága eltérhet a képernyős és kinyomtatott alakban
- a böngészők „Nyomtatási kép” menüjében első közelítésben könnyen ellenőrizhető egy weboldal nyomtatott formában való megjelenése

e) rugalmas doboz elrendezés

A rugalmas doboz (*flexible box* vagy röviden *FlexBox*) elrendezés egy CSS dobozmodell, melyben a rugalmas konténer gyermekei (*flex items*)

- elrendezhetők bármilyen haladási irányban (jobbra, balra, lefelé, felfelé)
- elhelyezhetők lineárisan egy (fő) tengely mentén, vagy több sorban egy másodlagos (keresztirányú) tengely mentén
- a rugalmas konténerhez vagy egymáshoz képest többféleképpen igazíthatók
- a rendelkezésre álló térnek megfelelően automatikusan változtathatják méretüket
- a vizuális megjelenítés független lehet a forráskódbeli sorrendiségtől.

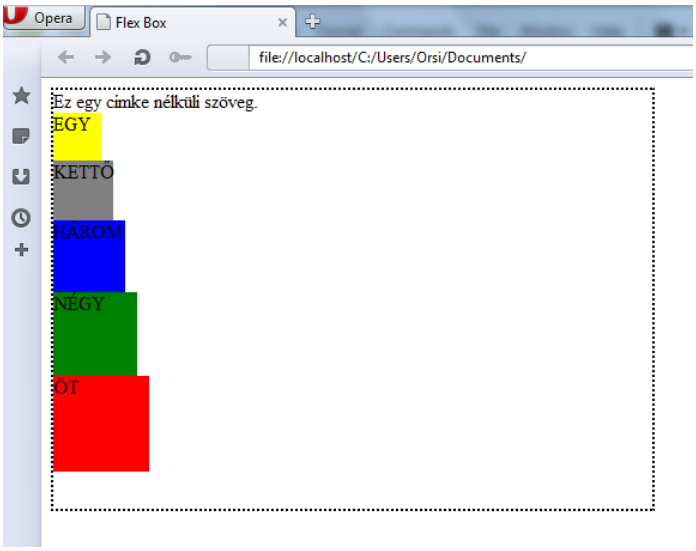
Megjegyzés:

- A *flexbox*-os elrendezés fontos szerepet fog játszani a különböző megjelenítő eszközökre készítendő, azonos kódolású weblapok esetén (*responsive design*)
- A *flexbox* megjelenítési módot az öt vezető böngésző már egyaránt alkalmazta, mikor a W3C változtatott a készülő szabványon. Az új verziót az Opera, Chrome és Firefox, továbbá az Internet Explorer 11-től értelmezi, a mobilokhoz és Safari-hoz –*webkit*- előtag szükséges.

A rugalmas elrendezés alkalmazási lehetőségeit egy *flex* konténerben elhelyezett öt *div* és egy egyszerű szöveg felhasználásával mutatjuk be (a *webkit*-es prefix-eket nem írtuk be).

Kiindulásként rugalmas elrendezés nélkül kódolva a *flex* konténerben lévő *div*-eket és a szöveget az alábbi, hagyományos kódsor és megjelenítés adódik:

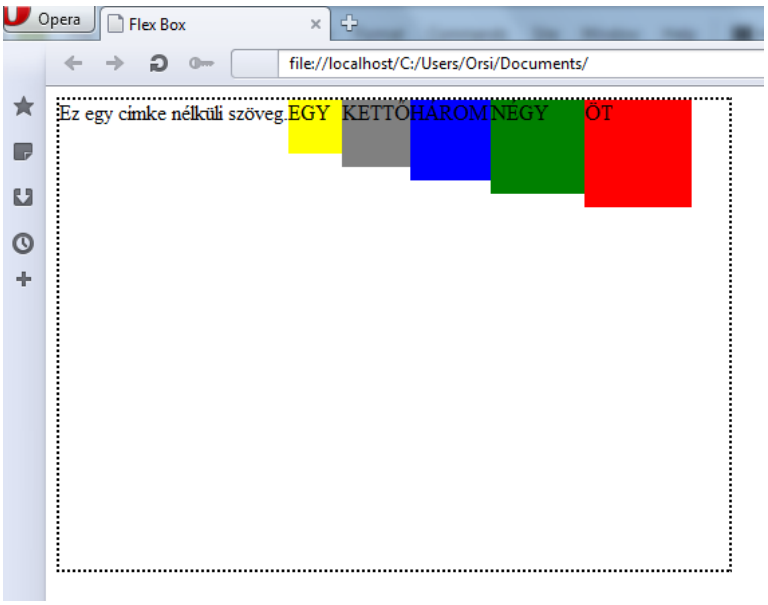
```
<!doctype html>
<html lang="hu">
    <head>
        <meta charset="utf-8">
        <title>Flex Box</title>
        <style>
            #flexbox { width:500px; height:350px; border:black dotted 2px; }
            #egy { width:40px; height:40px; background-color:yellow; }
            #ketto { width:50px; height:50px; background-color:grey; }
            #harom { width:60px; height:60px; background-color:blue; }
            #negy { width:70px; height:70px; background-color:green; }
            #ot { width:80px; height:80px; background-color:red; }
        </style>
    </head>
    <body>
        <div id="flexbox">
            Ez egy címke nélküli szöveg.
            <div id="egy">EGY</div>
            <div id="ketto">KETTŐ</div>
            <div id="harom">HÁROM</div>
            <div id="negy">NÉGY</div>
            <div id="ot">ÖT</div>
        </div>
    </body>
</html>
```



A rugalmas elrendezés lehet *flex* vagy *inline-flex*. Tekintettel arra, hogy a weblap rugalmas elrendezését kívánjuk megvalósítani, a továbbiakban a *flex* konténernél a megjelenítésre a már korábban ismert *display* tulajdonság új értékét, azaz a *display:flex* tulajdonság/érték párt – ugyancsak változatlan HTML kóddal - fogjuk mindig használni. A CSS:

```
<style>
   #flexbox { width:500px; height:350px; border:black dotted 2px; display:flex; }
   #egy { width:40px; height:40px; background-color:yellow; }
   #ketto { width:50px; height:50px; background-color:grey; }
   #harom { width:60px; height:60px; background-color:blue; }
   #negy { width:70px; height:70px; background-color:green; }
   #ot { width:80px; height:80px; background-color:red; }
</style>
```

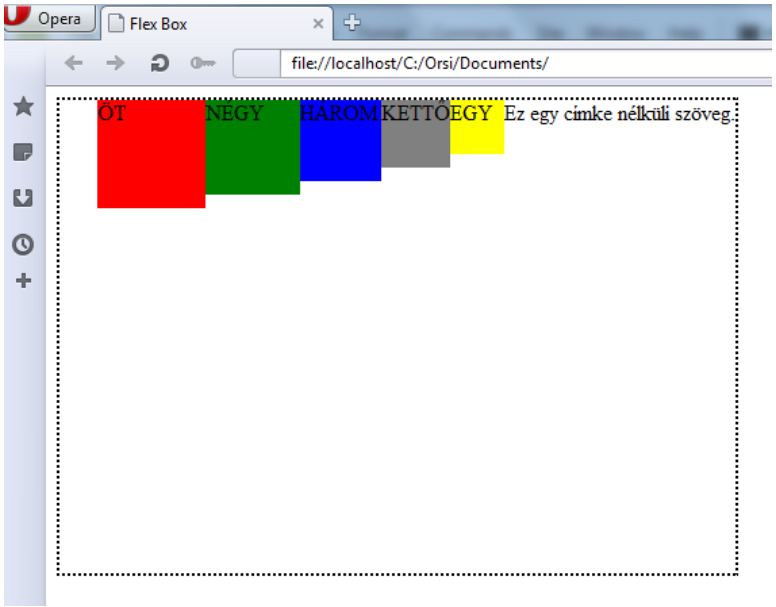
Az alapértelmezett rugalmas elrendezés megjelenítése pedig (a kódolatlan szöveget anonim *flex item*-ként kezeli a szabvány):



A flex konténerben sorban (*row*), oszlopban (*column*), megfordított sorban (*row-reverse*), vagy megfordított oszlopban (*column-reverse*) rendezhetők el a *flex item*-ek. Az ezt meghatározó tulajdonság neve *flex-direction*, az alapértelmezett értéke a *row*.

Egy fordított sorrendben, sorban elhelyezett elrendezés kódolása pl.:

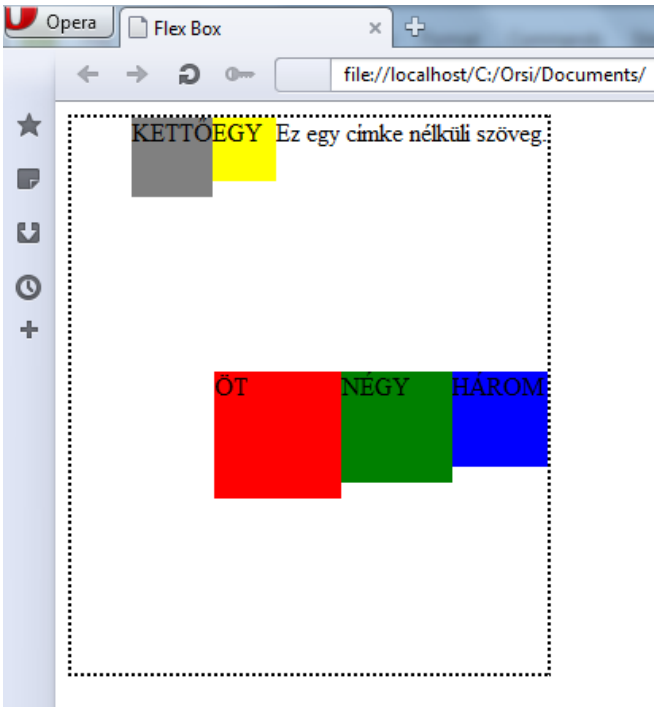
```
<style>
   #flexbox { width:500px; height:350px; border:black dotted 2px; display:flex; flex-
            direction:row-reverse; }
   #egy { width:40px; height:40px; background-color:yellow; }
   #ketto { width:50px; height:50px; background-color:grey; }
   #harom { width:60px; height:60px; background-color:blue; }
   #negy { width:70px; height:70px; background-color:green; }
   #ot { width:80px; height:80px; background-color:red; }
</style>
```



A *flex item*-ek sortörését a *flex-wrap* tulajdonság szabályozza. Alapértelmezett értéke a *nowrap*, mely esetben akkor sincsen sortörés (ill. oszloptörés), vagyis akkor is a főtengely mentén sorban (oszlopban) helyezkednek el a flex konténer gyermekei, ha nem férnek el a konténerben (túllógás). A *wrap* értéknél túllógás helyett sortörés/oszloptörés, azaz egy-soros/egyoszlopos elrendezés helyett többsoros/többoszlopos elrendezés jön létre.

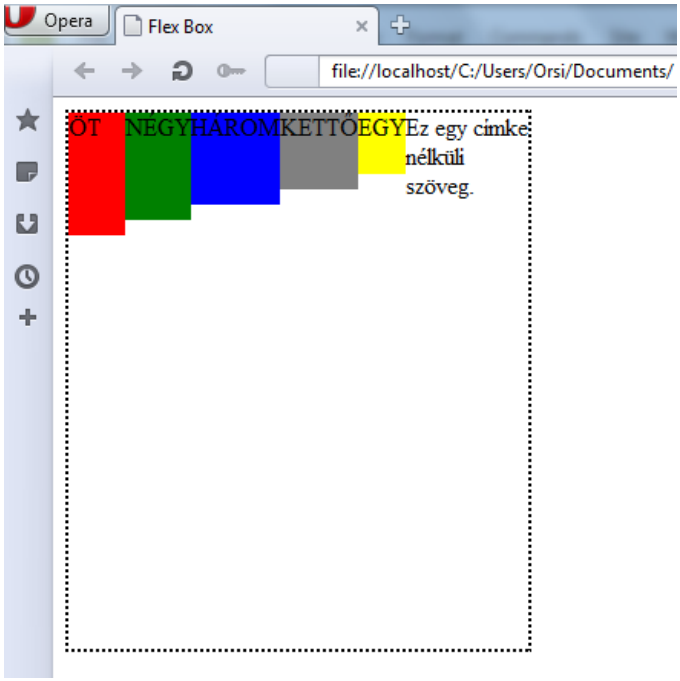
Az előző példát folytatva a flex konténer gyermekeinek automatikus sortörést kódolunk. Ahhoz, hogy látszódjon a hatása, megnöveljük az egyik gyermek méretét, így már nem férnek el a főtengely irányában (egy sorban), túllógás következne be:

```
<style>
   #flexbox { width:300px; height:350px; border:black dotted 2px; display:flex; flex-
            direction:row-reverse; flex-wrap:wrap; }
   #egy { width:40px; height:40px; background-color:yellow; }
   #ketto { width:50px; height:50px; background-color:grey; }
   #harom { width:60px; height:60px; background-color:blue; }
   #negy { width:70px; height:70px; background-color:green; }
   #ot { width:80px; height:80px; background-color:red; }
</style>
```



Amennyiben nem engedélyezett a sor/oszloptörés (*nowrap*), a további tulajdonságok függvényében alkalmazkodnak majd a *flex item*-ek a rendelkezésükre álló hely kitöltéséhez.

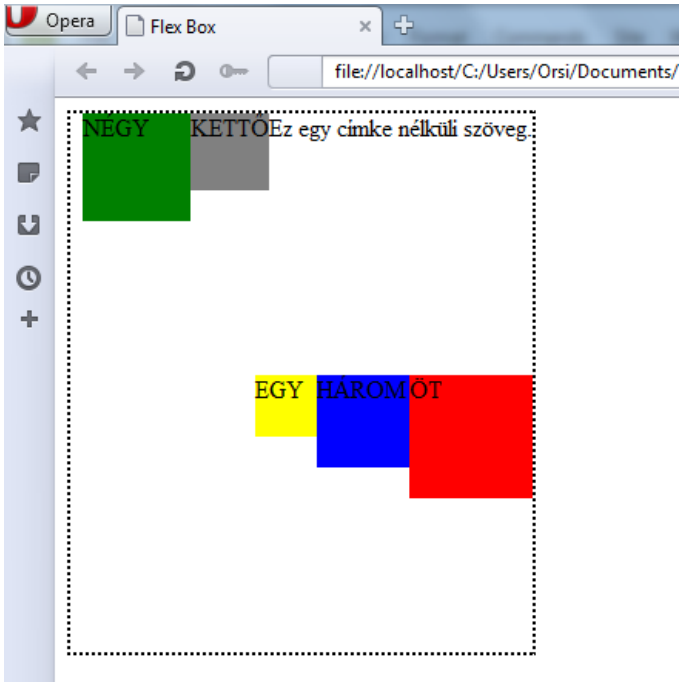
```
<style>
   #flexbox { width:300px; height:350px; border:black dotted 2px; display:flex; flex-
               direction:row-reverse; flex-wrap:nowrap; }
   #egy { width:40px; height:40px; background-color:yellow; }
   #ketto { width:50px; height:50px; background-color:grey; }
   #harom { width:60px; height:60px; background-color:blue; }
   #negy { width:70px; height:70px; background-color:green; }
   #ot { width:80px; height:80px; background-color:red; }
</style>
```



A *flex-direction* és *flex-wrap* tulajdonságok összevonhatóak *flex-flow* tulajdonságnév alatt, melynek az értékei a *flex-direction* és *flex-wrap* értékei, ebben a sorrendben, vessző nélkül felsorolva.

A *flex item*-ek megjelenítési sorrendje a kódolás sorrendjétől eltérően is meghatározható az *order* tulajdonsággal. Az *order* értékei pozitív és negatív egész számok lehetnek, a megjelenítési sorrend a növekvő számértékek szerint következik be:

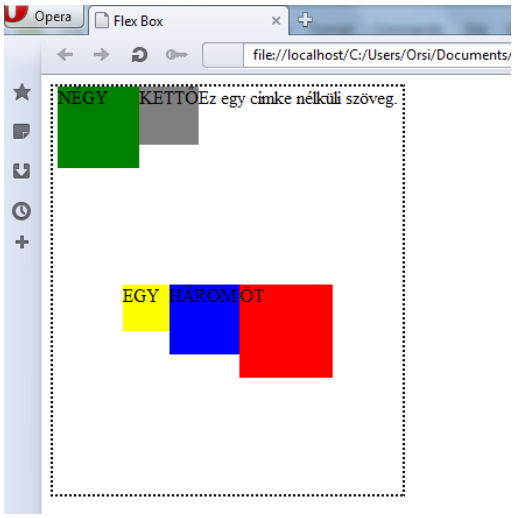
```
<style>
   #flexbox { width:300px; height:350px; border:black dotted 2px; display:flex; flex-
              direction:row; flex-wrap:wrap; }
   #egy { width:40px; height:40px; background-color:yellow; order:0; }
   #ketto { width:50px; height:50px; background-color:grey; order:-1; }
   #harom { width:60px; height:60px; background-color:blue; order:1; }
   #negy { width:70px; height:70px; background-color:green; order:-2; }
   #ot { width:80px; height:80px; background-color:red; order:2; }
</style>
```



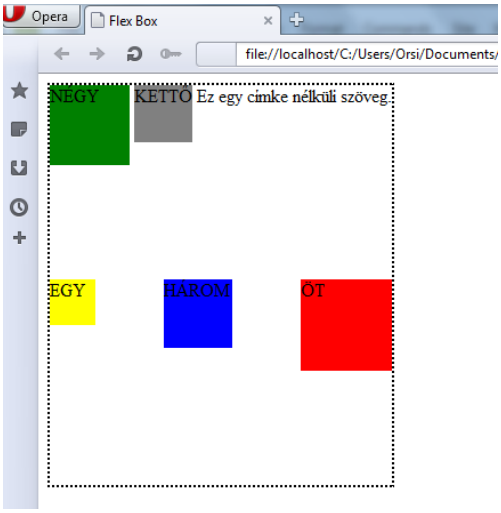
A *flex item*-ek főtengely mentén való elhelyezése a *justify-content* tulajdonsággal definiálható, melynek lehetséges értékei: *flex-start* (ez az alapértelmezett), *flex-end*, *center*, *space-between*, *space-around*.

A főtengely elejéhez (*flex-start*) vagy végéhez (*flex-end*), ill. a középre igazítás (*center*) kézenfekvő, lásd pl. az alábbi középre igazítást:

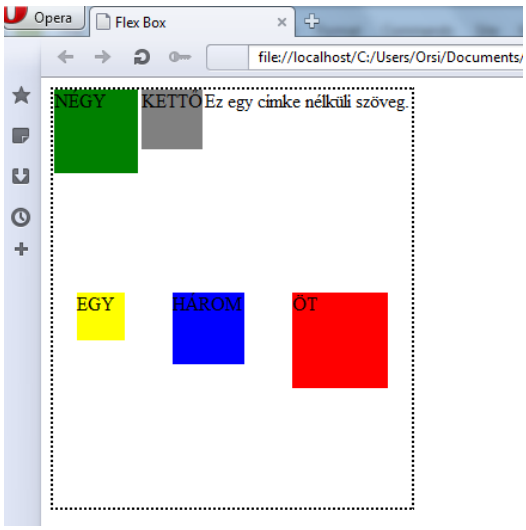
```
<style>
   #flexbox { width:300px; height:350px; border:black dotted 2px; display:flex; flex-
   direction:row; flex-wrap:wrap; justify-content:center; }
   #egy { width:40px; height:40px; background-color:yellow; order:0; }
   #ketto { width:50px; height:50px; background-color:grey; order:-1; }
   #harom { width:60px; height:60px; background-color:blue; order:1; }
   #negy { width:70px; height:70px; background-color:green; order:-2; }
   #ot { width:80px; height:80px; background-color:red; order:2; }
</style>
```



A ***justify-content:****space-between* a két szélső *flex item*-et a főtengely kezdő- és végpontjához rögzíti, a közbenső teret pedig úgy tölti ki, hogy egyenletes legyen a tereloszlás az *item*-ek között:



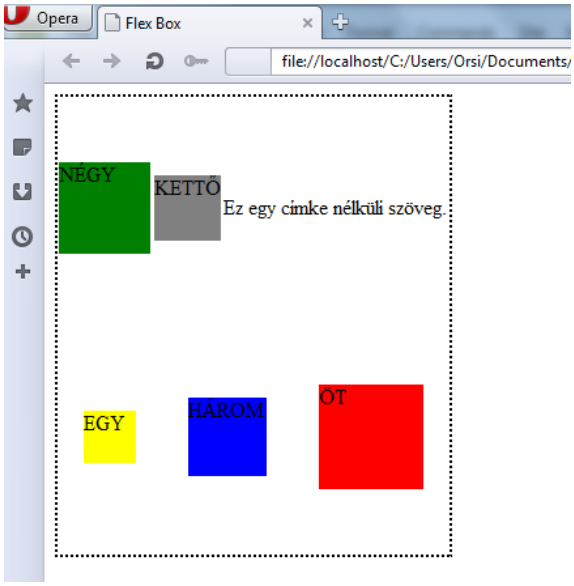
A ***justify-content:****space-around* egyenletes tereloszlást biztosít az *item*-ek között oly módon, hogy a főtengely végeitől a két szélső *item* fele olyan távolságra legyen, mint a közbenső térközök:



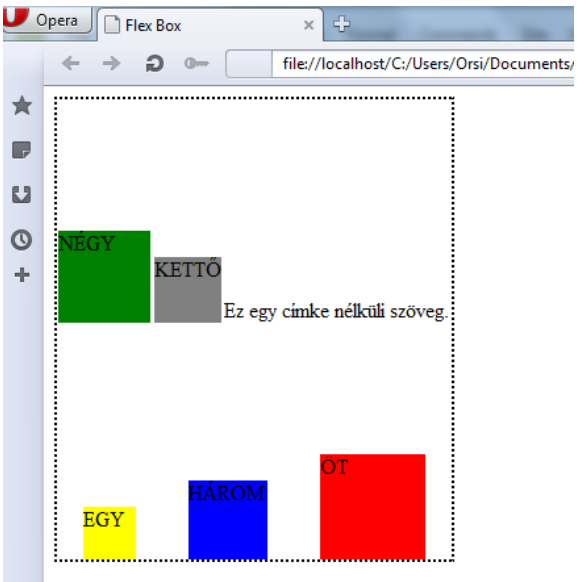
A főtengelyre merőleges *item*-igazítások az *align-items* tulajdonsággal definiálhatók, melynek lehetséges értékei: *stretch* (ez az alapértelmezett), *flex-start*, *flex-end*, *center*.

Center érték esetén pl. a főtengelyre merőleges irányban a két egyforma részre osztott konténer részeinek közepén helyezkednek el az *item*-ek:

```
<style>
   #flexbox { width:300px; height:350px; border:black dotted 2px; display:flex; flex-
         direction:row; flex-wrap:wrap; justify-content:space-around; align-items:center; }
   #egy { width:40px; height:40px; background-color:yellow; order:0; }
   #ketto { width:50px; height:50px; background-color:grey; order:-1; }
   #harom { width:60px; height:60px; background-color:blue; order:1; }
   #negy { width:70px; height:70px; background-color:green; order:-2; }
   #ot { width:80px; height:80px; background-color:red; order:2; }
</style>
```



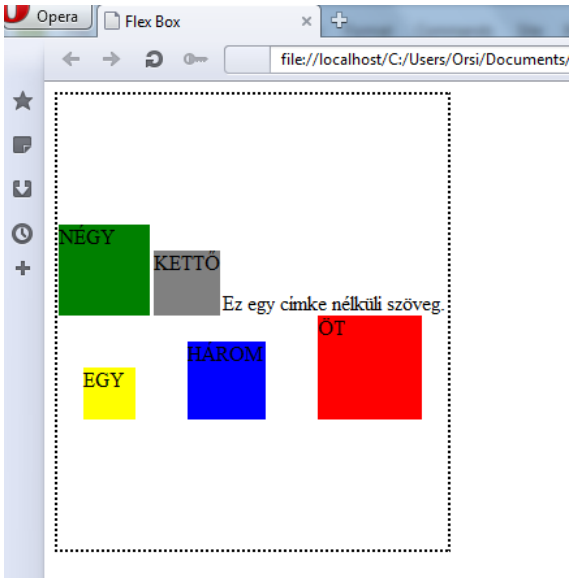
Az ***align-items:****flex-end* esetén a fenti kódolás megjelenítése (a kettéosztott konténer részeinek alsó éléhez igazodnak a *flex-item*-ek):



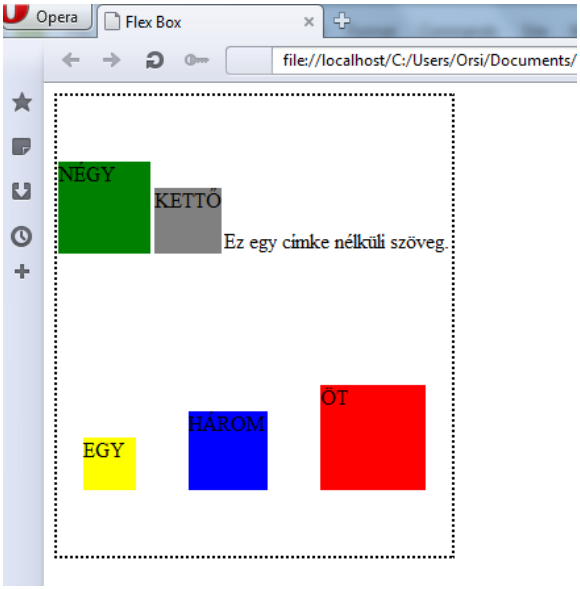
A főtengelyre merőleges további pozícionálási lehetőség valódítható meg az *align-content* tulajdonsággal, melynek alapértelmezett értéke a *stretch*, egyéb értékei a *center* és a *space-around*.

Példa az *align-content:center*-re:

```
<style>
   #flexbox { width:300px; height:350px; border:black dotted 2px; display:flex; flex-
            direction:row; flex-wrap:wrap; justify-content:space-around; align-items:flex-
            end; align-content:center; }
   #egy { width:40px; height:40px; background-color:yellow; order:0; }
   #ketto { width:50px; height:50px; background-color:grey; order:-1; }
   #harom { width:60px; height:60px; background-color:blue; order:1; }
   #negy { width:70px; height:70px; background-color:green; order:-2; }
   #ot { width:80px; height:80px; background-color:red; order:2; }
</style>
```



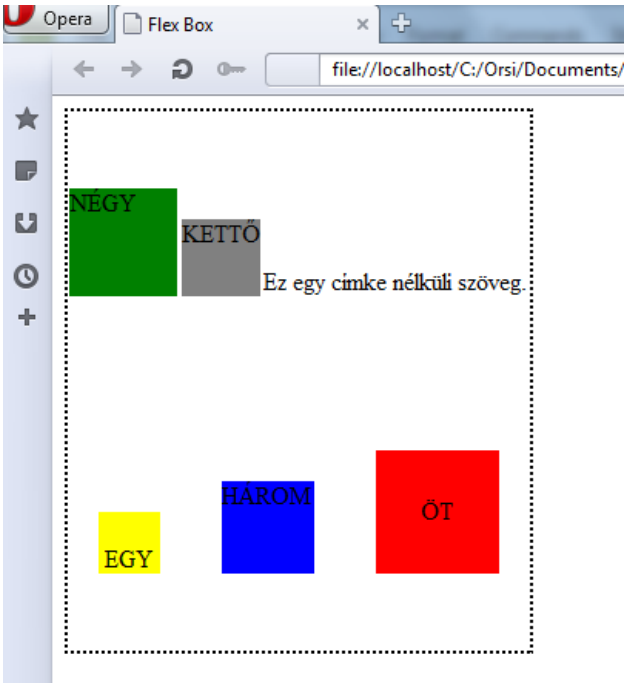
Példa az *align-content:space-around*-ra:



Az egyes *flex-item*-ek tartalmának elrendezését is elvégezhetjük olyan módon, hogy *flexbox*-oknak deklaráljuk őket, és aztán az előzőekben ismertetett kódolással hozzuk létre a kívánt elrendezést.

Az *„egy”* és *„öt”* *div*-ek szövegét középre-alulra ill.középre-középre pozícionálva a kódolás és a megjelenítés:

```
<style>
    #flexbox { width:300px; height:350px; border:black dotted 2px; display:flex; flex-direction:row; flex-wrap:wrap; justify-content:space-around; align-items:flex-end; align-content:space-around; }
    #egy { width:40px; height:40px; background-color:yellow; order:0; display:flex; justify-content:center; align-items:flex-end; }
    #ketto { width:50px; height:50px; background-color:grey; order:-1; }
    #harom { width:60px; height:60px; background-color:blue; order:1; }
    #negy { width:70px; height:70px; background-color:green; order:-2; }
    #ot { width:80px; height:80px; background-color:red; order:2; display:flex; justify-content:center; align-items:center; }
</style>
```



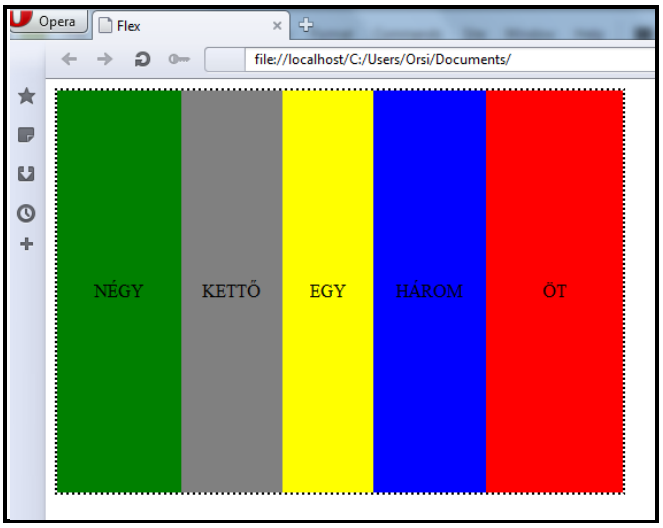
A *flexbox* elrendezés jelentőségét a különböző kijelzőméreteken való megjelenítéskor tovább növeli a kijelző méretváltozásakor előírható rugalmas változtatás lehetősége.

A *flex* összevont (shorthand) tulajdonság a *flex-grow*, *flex-shrink* és *flex-basis* tulajdonságokat fogja össze. A *flex-grow* és *flex-shrink* tulajdonságokkal definiálható, hogy *1px* kijelzőméretváltozás esetén hány *px*-el nőjjön ill. csökkenjen egy *item*-nak a mérete, míg a *flex-basis*-al az *item*-re egy referenciaméret adható meg.

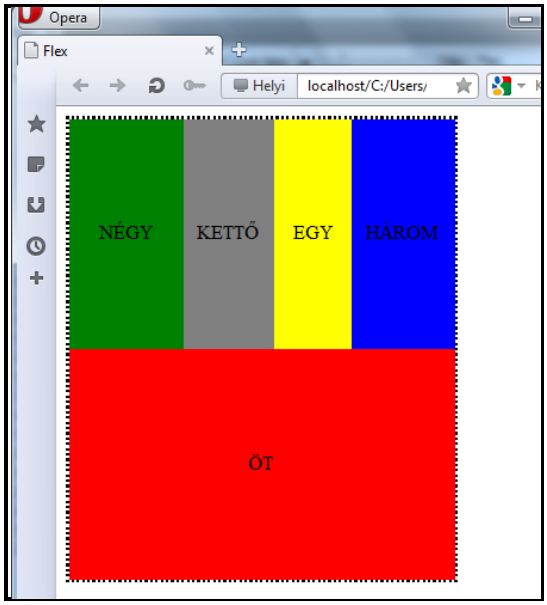
Vegyük pl. egy egyszerű esetet, egy weblap tartalmi részét (tehát a fejléccel, lábléccel, navigációs sávval most ne foglalkozzunk), melynek *div*-jei különböző szélességűek (40px, 50px, 60px, 70px, 80px), és a böngészőablak változtatásakor egyenletesen változnak. Az öt *div HTML*-kódja megegyezik a korábbiakkal, a CSS-kód pedig:

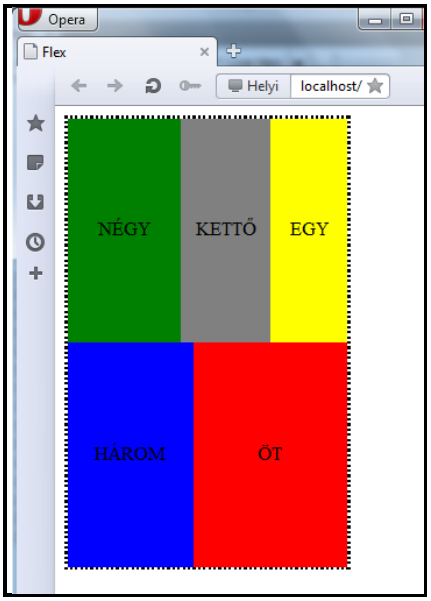
```
<style>
#flexbox { width:100%; height:350px; border:black dotted 2px; display:flex; flex-flow:row wrap;  }
#egy { background-color:yellow; order:0; display:flex; justify-content:center; align-items:center; flex:1 1 40px; }
#ketto { background-color:grey; order:-1; display:flex; justify-content:center; align-items:center; flex:1 1 50px; }
#harom { background-color:blue; order:1; display:flex; justify-content:center; align-items:center; flex:1 1 60px; }
#negy { background-color:green; order:-2; display:flex; justify-content:center; align-items:center; flex:1 1 70px; }
#ot { background-color:red; order:2; display:flex; justify-content:center; align-items:center; flex:1 1 80px; }
</style>
```

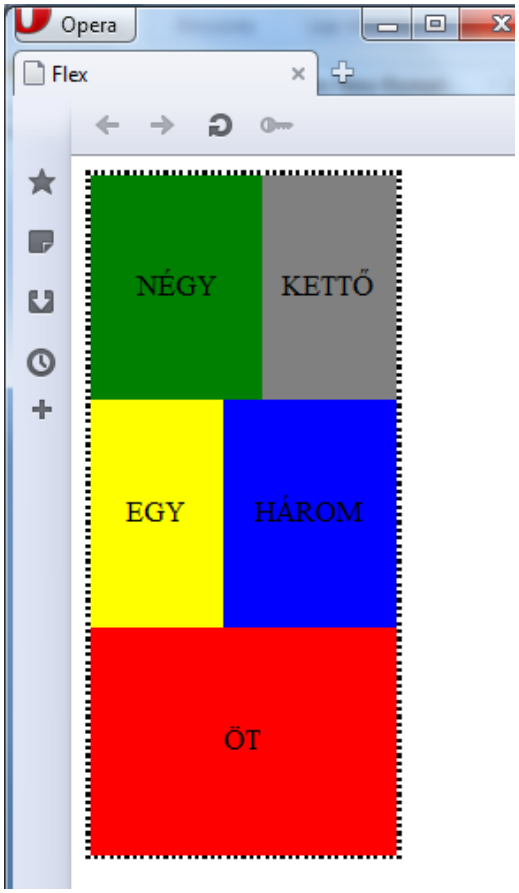
A tervezett képernyőméret esetén a megjelenítés:

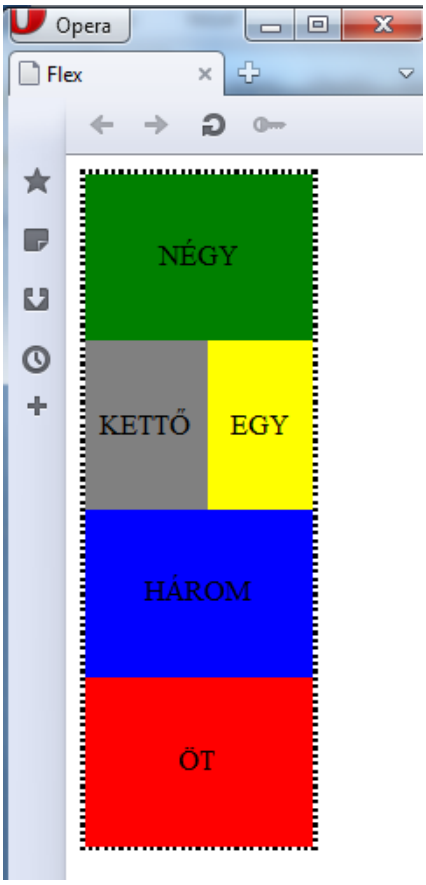


A böngészőablak szélességét csökkentve a weblap elrendezés az alábbiak szerint alakul:









Végül a hibrid (kevert) elrendezéssel kialakított weblapot alakítsuk ki *flexbox*-os technikával:

```
<!doctype html>
<html lang="hu">
    <head>
        <meta charset="utf-8">
        <title>flexbox elrendezés</title>
        <style>
            body, p, h1, h2, h3, table, img, ol, ul, form { margin:0; padding:0; }
                #main { width:100%; clear:left; }
                    header { background:blue; }
                    nav { background:red; }
                    #wrapper { display:flex; flex-flow:row wrap; }
                    #wrapper > article { background:yellow; flex:1 1 400px; order:0; }
                    #wrapper > aside   { background:green; flex:1 1 200px; order:1; }
                    footer { background:black; }
                  header, nav, footer { padding:10px; text-align:center; color:white; clear:left;
width:100%; height:40px; }
        </style>
    </head>
    <body>
        <div id="main">
            <header>HEADER = FEJLÉC</div></header>
            <nav>NAV = NAVIGÁCIÓS SÁV</nav>
```

```
        <div id="wrapper">
            <aside><h1>"aside" oszlop</h1><p>Proin sed elit felis, in facilisis arcu.
Aliquam malesuada, magna quis eleifend pharetra, nunc nunc pretium sapien, mattis
consequat nulla elit a nulla. Nullam tincidunt mattis odio. Morbi dapibus sem sit amet nibh
lobortis sed interdum est lacinia. Morbi inconvallis lectus. Ut ac hendrerit neque. Nullam
pellentesque, arcu non mattis egestas, arcu tortor convallis orci, ac ultricies urna lacus at
nisi. Curabitur dignissim nunc quis libero convallis feugiat. Mauris consequat, lorem a
feugiat aliquet, nibh mauris lobortis risus, in porta felis augue attortor. Vivamus ullamcorper
interdum mattis.</p></aside>
            <article><h1>"article" oszlop</h1><p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Sed tempus augue in erat dignissim accumsan. Suspendisse eu vestibulum dui.
Phasellus tortor mauris, accumsan nec dapibus a, sodales nec ante. Duis sollicitudin tortor id
dui convallis vel sagittis augue ornare. Aenean quam urna, tempor in ultrices id, interdum et
sem. Aliquam sollicitudin faucibus magna ut porta. Etiam feugiat venenatis mauris, sed
posuere erat mollis eu. Ut sit amet leo nisi, quis aliquam felis. Sed sit amet urna vel urna
tincidunt elementum eget vitae dolor. Nunc vitae imperdiet augue. Proin ultrices libero vitae
enim lacinia consequat.</p>
                <p>Vivamus eu lorem non tortor euismod laoreet in ut purus. Morbi laoreet
mollis enim, in bibendum eros consectetur nec. Vestibulum ultricies adipiscing lectus, vel
condimentum neque facilisis non. In hac habitasse platea dictumst. In pulvinar augue ac nisi
placerat et varius arcu imperdiet. Nulla facilisi. Aenean quis odio erat, nec accumsan lectus.
Curabitur gravida, felis ut vestibulum adipiscing, dui elit rutrum nulla, nec interdum ipsum
enim id tortor. Sed odio nisi, sodales vel cursus a, porta eu ante. Fusce nec arcu rhoncus eros
commodo fringilla. Suspendisse nec pellentesque turpis. Class aptent taciti sociosqu ad litora
torquent per conubia nostra, per inceptos himenaeos. Vestibulum a arcu eu arcu commodo
dapibus scelerisque ac ligula. Duis neque lacus, mollis a faucibus eget, consectetur eget
ligula.</p>
                <p>Quisque consectetur, mauris a iaculis eleifend, metus dolor consequat
enim, eu consec-tetur ante lorem sed ligula. Vivamus in suscipit leo. Nullam consectetur
vulputate justo nec viverra. Sed porttitor nulla ac ligula molestie euismod. Proin laoreet
ullamcorper tempus. </p></article>
        </div>
            <footer>FOOTER = LÁBLÉC</footer>
        </div>
    </body>
</html>
```

Megjegyzés: a *flexbox*-os technika is ötvözhető szükség esetén a *media* jellemzővel.

4. SVG 2

A vektorképeket matematikai képlettel megadható elemek alkotják, melyek a paraméterek változtatásával minőségromlás nélkül átméretezhetők. A *web*-en legelterjedtebben alkalmazott, kétdimenziós grafikákat leíró ilyen nyelv az SVG (Scalable Vector Graphics = átméretezhető vektorgrafika). Ezt a HTML-el és CSS-el jól együttműködő, HTML dokumentumba ágyazva és külön is használható, nyílt forráskódú és szabad felhasználású jelölőnyelvet valamennyi böngésző (az Internet Explorer-ek esetében csak a 9-es verzió és az ennél újabbak) jól értelmezi, és ugyancsak egyszerű editorban kódolható.

W3C 1998-ban alakította meg az SVG WG-t (WG=Working Group = munkacsoport), 2001 szeptemberére elkészült az SVG 1.0 ajánlás, majd 2003 januárjára az SVG 1.1 ajánlás első kiadása. Többéves szünetet követően (mely során elkezdték majd abbahagyták az 1.2 verzió fejlesztését) 2011 augusztusában elfogadták az SVG 1.1 ajánlás második kiadását, és jelenleg zajlik az SVG 2 fejlesztése.

Ismertetésünkben HTML dokumentumba foglalt SVG-t mutatunk be, melynek nyelvtanában csak annyi lesz a HTML-hez képest az eltérés, hogy a páratlan címkéket is le kell (az XML szabályok miatt) zárni. Dokumentumaink szerkezete tehát minden alkalommal így fog kinézni (a *head*-et és *body*-t nem írom ki):

```
<!doctype html>
<html lang="hu">
   <svg width="……" height="……">
      .
   <…>……</…>            ( a páros címkék esetében )
      .
   <…… />                ( a páratlan címkék esetében )
      .
   </svg>
</html>
```

A jellemzők és értékek többsége a CSS tulajdonságokból és értékekből vehető át. Az SVG grafikák helyzete *xy* koordinátarendszerben definiálható, melynek kezdőpontja a CSS-nél látottakkal megegyezően a grafikát tartalmazó doboz (*container box*) bal felső sarka, *x* és *y* pozitív értékei a bal saroktól a dobozban jobbra ill. lefelé irányt definiálnak – viszont automatikusan nem képződik a tartalomhoz terület, ezért mindig meg kell adni *width* és *height* értékeket az <svg> nyítócímkében a *content area* számára!

Három grafikatípus megengedett az SVG-ben (melyekhez különböző vizuális effektek is rendelhetők): alakzat, szöveg és kép.

4.1. Alakzatok

A különböző alakzatok kontúrját alkotó vonalakat a páratlan – ezért önmagában lezárást is tartalmazó - *<path />* címkével (*path* = út/szakasz) definiáljuk. Jellemzője a szakasz jellegét és elhelyezését definiáló *d="……."*, melynek értékei a *path* elem *paraméterei*, azaz *utasítások* és *xy koordináták.*

1.1. Egyenes vonalak

Egy egyenest a kezdő és végpontjával lehet definiálni. Egy kezdőpont létrehozása az *M* utasítás (M = moveto) és a kezdőpont *x,y* koordinátáinak megadásával, egy egyenes vonal meghúzása pedig az *L* utasítás (L = lineto)-al és a végpont *x,y* koordinátáinak a megadásával

történik. Az origo-ból (*x,y = 0*) ferdén lefelé az *x=150, y=100* pontba húzott egyenes kódolása tehát:

```
<!doctype html>
<html lang="hu">
   <svg width="200" height="200">
      <path d="M0 0 L150 100" />
   </svg>
</html>
```

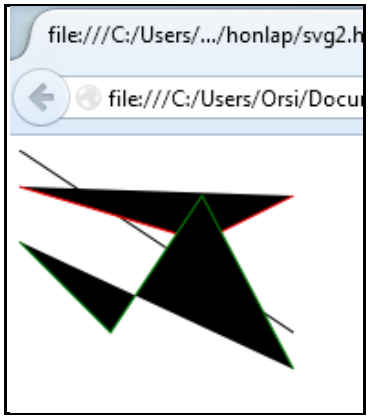
Figyelem! Bár a formázással majd később foglalkozunk, két dolgot már az elejétől kezdve ismerni és alkalmazni kell. Az egyik: valamennyi *path* vonal/szakasznak a színét a *stroke="…."* jellemzővel és a CSS-ből ismert színértékekkel lehet megadni – csakhogy az alapértelmezett érték a *none* , így ha nem definiálunk valamilyen színt, akkor semmit nem látunk a kódolt grafikából. Tehát ahhoz, hogy a fenti egyenes látszódjon, színt kell kódolni hozzá. Pl. fekete egyenes vonal esetén:

```
<!doctype html>
<html lang="hu">
   <svg width="200" height="200">
      <path d="M0 0 L150 100" stroke="black" />
   </svg>
</html>
```

Különböző irányú és hosszúságú egyenesek újabb *L* utasítások és végpontok definiálása révén a megelőző végpontokhoz hozzáfűzhetők, így pl. 2 (piros), ill. 3 (zöld) egyenes szakaszból álló nyílt alakzat kódolása (meghagyva a fekete első vonalat is):

```
<!doctype html>
<html lang="hu">
  <svg width="200" height="200">
    <path d="M0 0 L150 100" stroke="black" />
    <path d="M0 20 L100 50 L150 25" stroke="red" />
    <path d="M0 50 L50 100 L100 25 L150 120" stroke="green" />
  </svg>
</html>
```

A kódolás az SVG-vel ismerkedőknek valószínüleg nem várt megjelenítést eredményez:

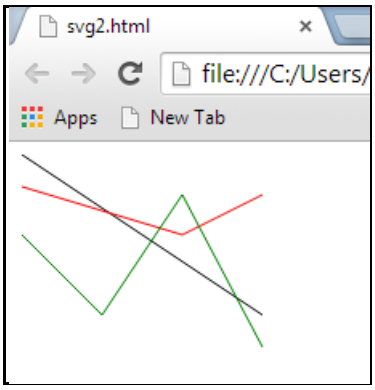


Figyelem! A formázással kapcsolatos másik dolog (az első a *stroke* volt), amit már a legelején tisztázni kell, hogy az alakzatok kitöltését a *fill="…."* jellemzővel és a CSS-ből ismert színértékekkel lehet megadni, melynek alapértelmezett értéke „*black*", és nemcsak a zárt alakzatokat tölti ki feketével, hanem a nyílt alakzatokból is (kitöltés szempontjából)

zártakat hoz létre a nem egy vonalra eső pontok összekötésével. Így ha kitöltés nélküli alakzatot akarunk, definiálni kell, hogy *fill="none"*. Ennek egy egyenes vonal esetében nincsen jelentősége – hiszen (egydimenziós elemről lévén szó) nincs mit kitölteni – de az egységes kódolás érdekében itt is célszerű a *fill* értéket megadni.

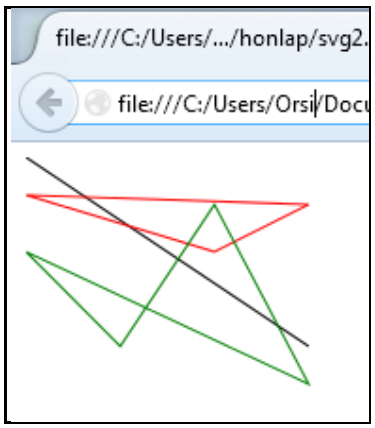
Ha tehát ténylegesen a három vonalat akarjuk látni, akkor a kódolás és a megjelenítés:

```
<!doctype html>
<html lang="hu">
  <svg width="200" height="200">
   <path d="M0 0 L150 100" fill="none" stroke="black" />
   <path d="M0 20 L100 50 L150 25" fill="none" stroke="red" />
   <path d="M0 50 L50 100 L100 25 L150 120" fill="none"
         stroke="green" />
  </svg>
</html>
```



Az utolsó pontot és a kezdőpontot összekötő, alakzatbezáró egyenest a *Z* utasítással (Z= closepath) kódolhatjuk – itt mind a kezdőpont, mind a végpont adott, tehát nincsen szükség koordináták definiálására. A három vonalunkat ilyen módon zárt alakzattá alakítva:

```
<!doctype html>
<html lang="hu">
  <svg width="200" height="200">
    <path d="M0 0 L150 100 Z" fill="none" stroke="black" />
    <path d="M0 20 L100 50 L150 25 Z" fill="none" stroke="red"
          />
    <path d="M0 50 L50 100 L100 25 L150 120 Z" fill="none"
          stroke="green" />
  </svg>
</html>
```



Az egyszerű egyenes vonal esetében nyilvánvalóan nem lehetett hatása az alakzatbezárásnak.

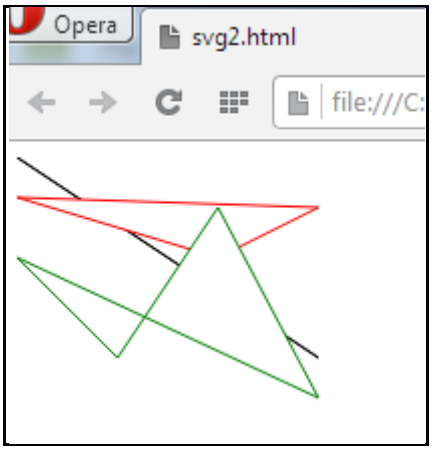
A kitöltésre visszatérve vizsgáljunk meg még két esetet:

a) a kitöltés megegyezik a háttérszínnel

b) különböző kitöltések átfedik egymást

a) Ha a háttérszínt, azaz a fehéret kódoljuk be, akkor:

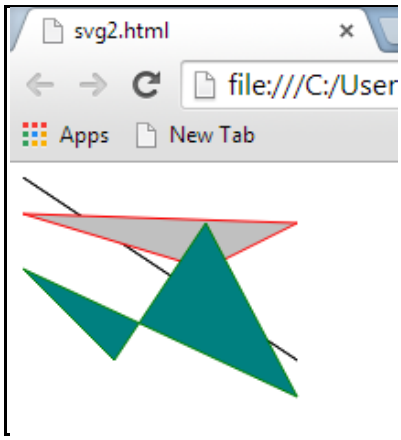
```
<!doctype html>
<html lang="hu">
  <svg width="200" height="200">
     <path d="M0 0 L150 100" fill="white" stroke="black" />
     <path d="M0 20 L100 50 L150 25 Z" fill="white"
           stroke="red" />
     <path d="M0 50 L50 100 L100 25 L150 120 Z" fill="white"
           stroke="green" />
  </svg>
</html>
```



A megjelenítésből jól látszik, hogy a „nincsen kitöltés” és „háttérszínnel kitöltés” eltérő eredményt ad; az első esetben nincsen, a második esetben van takarás.

b) Különböző kitöltések átfedésekor keletkező takarás esetén a kódolás sorrendje a döntő:

```
<!doctype html>
<html lang="hu">
   <svg width="200" height="200">
     <path d="M0 0 L150 100 Z" stroke="black" fill="black" />
     <path d="M0 20 L100 50 L150 25 Z" fill="silver"
           stroke="red" />
     <path d="M0 50 L50 100 L100 25 L150 120 Z" fill="teal"
           stroke="green" />
   </svg>
</html>
```



Az ismert szabály szerint a későbbi kód felülírja a korábbit (ez egyébként a fehér hátteres példán is látszik, csak nem ennyire szemléletesen).

Az alakzatnak HTML-ben „Átfedések” címet adva ellenőrizzük az SVG koordináta-rendszer origójának viselkedését:

```
<!doctype html>
<html lang="hu">
   <h1>Átfedések</h1>
   <svg width="200" height="200">
     <path d="M0 0 L150 100 Z" stroke="black" fill="black" />
     <path d="M0 20 L100 50 L150 25 Z" fill="silver"
           stroke="red" />
     <path d="M0 50 L50 100 L100 25 L150 120 Z" fill="teal"
           stroke="green" />
   </svg>
</html>
```

A cím nélküli kód esetében a weblap törzsének (*body*) origója volt egyben a grafikát befoglaló doboz origója is, míg a címsoros kódnál a *h1* blokkszintű elem alatt az SVG blokk „lejjebb csúszott”, de a grafika változatlan maradt - tehát csakugyan a befoglaló doboza bal felső sarka az origója, nem pedig a dokumentum origójától kell méretezni a grafikát.

A *path* elemek *d* jellemzőjében lévő utasítások megadhatók **abszolút értékkel** (ezeket jelöljük nagybetűkkel), és **relatív értékkel** (ezeket kisbetűkkel jelöljük). Az abszolút érték az origótól mért *x,y* koordinátákat, míg a relatív érték az aktuális kiindulási pont és aktuális végpont közötti x,y érték***változást*** fejezi ki.

A három vonalat eddig nagybetűs utasításokkal, tehát abszolút értékekkel kódoltuk:

```
<!doctype html>
<html lang="hu">
 <svg width="200" height="200">
  <path d="M0 0 L150 100" fill="none" stroke="black" />
  <path d="M0 20 L100 50 L150 25" fill="none" stroke="red" />
  <path d="M0 50 L50 100 L100 25 L150 120" fill="none"
        stroke="green" />
 </svg>
</html>
```

Ugyanez a grafika kisbetűs utasításokkal, tehát relatív értékekkel kódolva:

```
<!doctype html>
<html lang="hu">
 <svg width="200" height="200">
  <path d="M0 0 l150 100" fill="none" stroke="black" />
  <path d="M0 20 l100 30 l50 -25" fill="none" stroke="red" />
  <path d="M0 50 l50 50 l50 -75 l50 95" fill="none"
        stroke="green" />
 </svg>
</html>
```

Mert: l_x=150-0 l_y=100-0 | l_x=100-0 l_y=50-20 | l_x=150-100 l_y=25-50 | l_x=50-0 l_y=100-50 | l_x=100-50 l_y=25-100

Ameddig az egymást követő utasítások megegyeznek (beleértve az abszolút és relatív értékmegadást is), addig az első után az utasítások elhagyhatóak, és elegendő csak az abszolút ill. relatív értékeket megadni. Tehát az eddig használt három vonal így is kódolható:

```
<!doctype html>
<html lang="hu">
 <svg width="200" height="200">
  <path d="M0 0 L150 100" fill="none" stroke="black" />
  <path d="M0 20 L100 50 150 25" fill="none" stroke="red" />
  <path d="M0 50 L50 100 100 25 150 120" fill="none"
        stroke="green" />
 </svg>
</html>
```

```
<!doctype html>
<html lang="hu">
 <svg width="200" height="200">
  <path d="M0 0 l150 100" fill="none" stroke="black" />
  <path d="M0 20 l100 30 50 -25" fill="none" stroke="red" />
  <path d="M0 50 l50 50 50 -75 50 95" fill="none"
        stroke="green" />
 </svg>
</html>
```

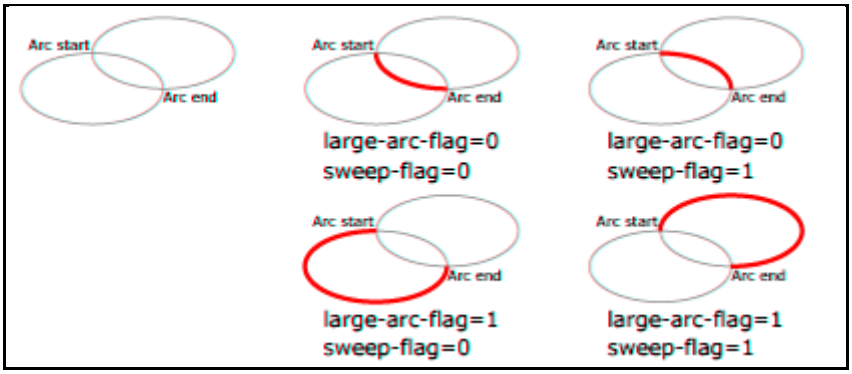
A vízszintes és függőleges vonalra külön utasítások H,h (horizontal) és V,v (vertical) abszolút és relatív pozicionálással is alkalmazhatóak – ezek az általános L,l utasítás egyszerűsített alakjai, ahol a vízszintesnél az *y=0*, a függőlegesnél az *x=0*. Például egy vízszintes egyenes relatív értékkel megadva lehet egyszerűen ***h100*** egy függőleges pedig ***v50***.

1.2. Görbék

Pontok görbékkel való összekötése ellipszis- ill. Bezier-görbe szakaszokkal kódolható.

a) Ellipszis szakaszok (speciális esete a körszakasz is)

Két pontot (*arc start* ill. *arc end*) vízszintes főtengelyű ellipszis szakaszokkal és adott sugárral 4-féleképpen lehet összekötni: a nagyobbik íven pozitív vagy negatív szögek irányába haladva, ill. a kisebbik íven pozitív vagy negatív szögek irányába haladva. Ábrázolva:

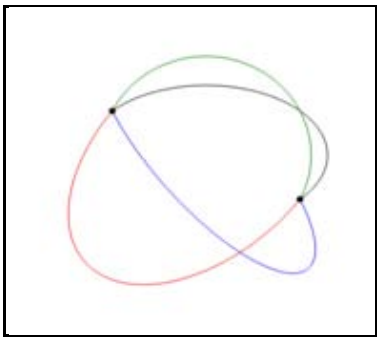


A kódolásban a nagyobbik ív *1* –el (logikai „igaz”), a kisebbik ív *0-val* (a nagyobbik ív logikailag „hamis”) definiálható. Ezzel analóg módon ha a haladási szög pozitív, akkor *1* , ellenkező esetben *0* adandó meg. A két érték vesszővel elválasztva kódolandó – az ellipszis szakasz fókuszpontjait az *arc start*, *arc end* koordinátái és az ellipszis két sugarának általunk definiált mérete ismeretében automatikusan kiszámolja az SVG.

Tekintettel arra, hogy tetszőleges ellipszis sugárméretek és főtengelyirány is definiálható - az utóbbit az *x*-tengellyel bezárt szöggel kell megadni (a fenti ábrán ez *0* volt) – két pont között végtelen az ellipszis szakaszok variációs lehetőségeinek a száma.

A két pontot összekötő, ellipszis szakasszal definiált görbék utasítása az *A* (A= arc =ív), melyet a kezdőpont, az *x,y* irányú sugárméretek, a főtengelyirány, a 4 lehetséges útvonal közüli választás, és a végpont paraméterei követnek.

```
<!doctype html>
<html lang="hu">
 <svg width="500" height="500">
   <path d="M300 250 A90 55 0 1,0 100 150" stroke="black"
        fill="none" />
   <path d="M300 250 A55 90 45 0,1 100 150" stroke="red"
        fill="none" />
   <path d="M300 250 A75 75 90 0,0 100 150" stroke="green"
        fill="none" />
   <path d="M300 250 A30 10 -130 1,1 100 150" stroke="blue"
        fill="none" />
 </svg>
</html>
```

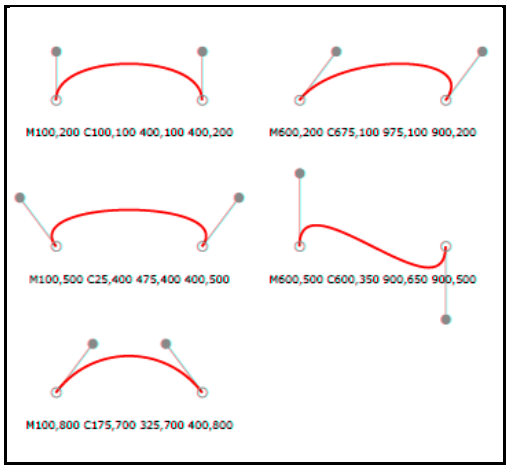


b) Bezier-görbe szakaszok (négyzetes és kvadratikus)

A ***négyzetes*** Bezier-görbe a kezdő- és végpontjával, továbbá két ellenőrzőponttal definiálható, utasítása *C* (C=cubic=négyzetes). Például:

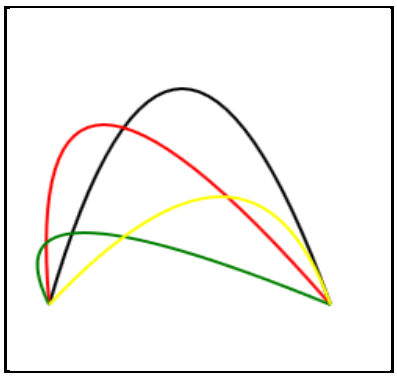
```
<!doctype html>
<html lang="hu">
 <svg width="600" height="500">
   <path d="M100 200 C100 100 400 100 400 200" stroke="red"
          fill="none" />
 </svg>
</html>
```

Az ellenőrzőpontok változtatásával a négyzetes Bezier-görbe alakja módosítható:



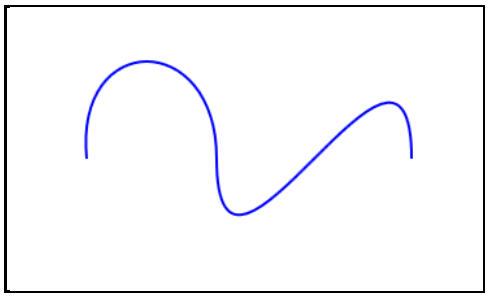
A ***kvadratikus*** Bezier-görbe a kezdő- és végpontjával, továbbá egy ellenőrzőponttal definiálható, utasítása *Q* (Q=quadratic=kvadratikus). Az ellenőrző pont változtatásával a kvadratikus Bezier-görbe alakjának változása:

```
<!doctype html>
<html lang="hu">
 <svg width="300" height="300">
  <path d="M20 200 Q100 -100 200 200" stroke="black"
        fill="none" />
  <path d="M20 200 Q0 -50 200 200" stroke="red" fill="none" />
  <path d="M20 200 Q-25 100 200 200" stroke="green"
        fill="none" />
  <path d="M20 200 Q150 50 200 200" stroke="yellow"
        fill="none" />
 </svg>
</html>
```



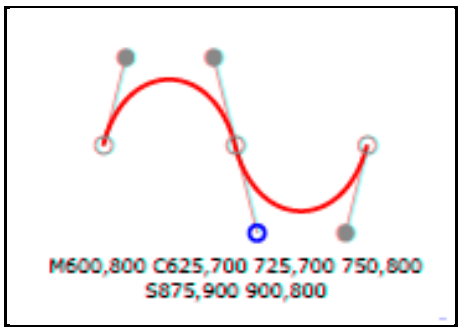
A görbék (az egyenesekhez hasonlóan) láncba kapcsolhatók, például két négyzetes Bezier-görbe (ami *polybezier*-t alkot) összekapcsolása:

```
<!doctype html>
<html lang="hu">
 <svg width="400" height="300">
  <path d="M100 200 C90 100 200 100 200 200 C200 350 350 50
           350 200" stroke="blue" fill="none" />
 </svg>
</html>
```

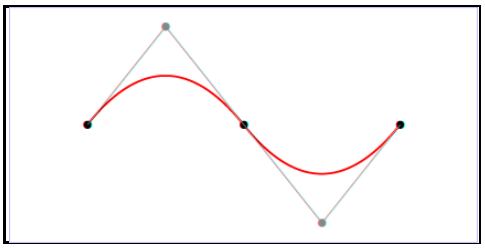


A gyakorlatban nagyon hasznos a **folyamatosan illeszkedő** (*smooth curveto* vagy *shorthand*-nek nevezett) esetet megvalósító kódolási lehetőség (az előző, fenti görbeillesztés csak szemmérték alapján, és nem precízen, a kódra bízva történt).

Két négyzetes Bezier-görbe úgy illeszkedik folyamatosan, ha a csatlakozási pontjukban a második görbe első ellenőrzőpontja az előző görbe második ellenőrzőpontjának a csatlakozási pontra vonatkoztatott tükörképe (vagyis a görbék érintői a csatlakozási pontban egybeesnek). A folyamatosan illeszkedő négyzetes Bezier-görbe utasítása ***S*** (relatív értékek esetén ***s***), definiálásához a görbe második ellenőrzőpontja és a végpontja szükséges (hiszen a kezdőpont és az első ellenőrzőpont az előző görbe végpontjával és annak második ellenőrző-pontjával megegyező).



Két kvadratikus Bezier-görbe úgy illeszkedik folyamatosan, ha a csatlakozási pontjukban a második görbe ellenőrzőpontja az előző görbe ellenőrzőpontjának a csatlakozási pontra vonatkoztatott tükörképe (vagyis a görbék érintői a csatlakozási pontban egybeesnek). A folyamatosan illeszkedő kvadratikus Bezier-görbe utasítása ***T*** (relatív értékek esetén ***t***), definiálásához a görbe végpontja szükséges (hiszen a kezdőpont és az ellenőrzőpont az előző görbe végpontjával és ellenőrzőpontjával megegyező).



Az egyenes szakaszokkal és görbékkel kódolt grafikának egy egyszerű alkalmazása a kördiagram. Alsó (nagy, ¾ körös) körszelete:

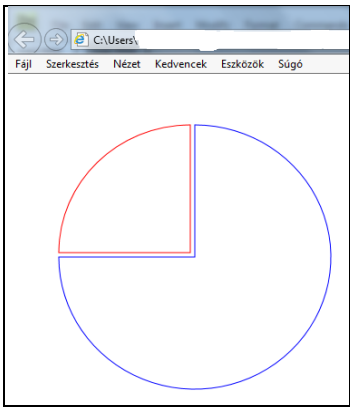
```
<path d="M200 200 h-150 a150,150 0 1,0 150 -150 Z"
      stroke="blue" fill="none" />
```

Felső (kicsi, 1/4 körös) körszelete:

```
<path d="M195 195 v-145 a145,145 0 0,0 -145 145 Z"
      stroke="red" fill="none" />
```

A teljes kördiagram kódolása és a megjelenítés:

```
<!doctype html>
<html lang="hu">
  <svg width="400" height="400">
    <path d="M200 200 h-150 a150,150 0 1,0 150 -150 Z"
          stroke="blue" fill="none" />
    <path d="M195 195 v-145 a145,145 0 0,0 -145 145 Z"
          stroke="red" fill="none" />
  </svg>
</html>
```



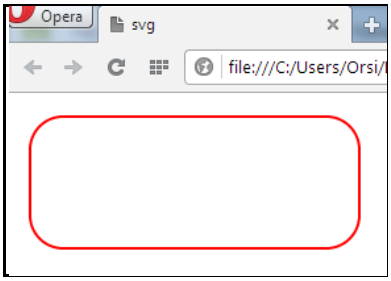
1.3. Alapvető alakzatok

Az alapvető alakzatok (*basic shapes*) a négyszögek (beleértve a lekerekített sarkúakat is), körök, ellipszisek, egyenes vonalak, egyenes vonalak alkotta nyílt alakzatok (*polyline*),

és az egyenes vonalakból alkotott zárt alakzatok (*polygon*). Matematikailag ezek az azonos alakzatot létrehozó *path* elemmel ekvivalensek, csak egyszerűbben kódolhatók.

1.) A *rect* **négyzet** elem a bal felső sarkának a helyzetével (*x,y*), a szélességével (*width*) és a magasságával (*height*) specifikálható. Az aktuális koordinátarendszerhez (x,y tengelyekhez) igazított *x,y* értékek a bal felső saroktól számítandók. Sarkainak lekerekítése a lekerekítési sugarak mint *rx* és *ry* jellemzők értékeivel adható meg. Ha csak egy érték van specifikálva, vagy a két érték közül az egyik hibás, akkor mindkét lekerekítési sugarat azzal azonosnak veszi a böngésző. Ha *rx* nagyobb a *width* felénél, ill. *ry* nagyobb a *height* felénél, akkor a fele értékre veszi vissza a sugarat a böngésző.

```
<!doctype html>
<html lang="hu">
  <svg width="300" height="150">
    <rect x="10" y="10" width="250" height="100" rx="25"
          fill="none" stroke="red" />
  </svg>
</html>
```

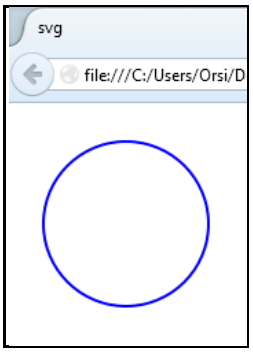


Megjegyzés: Egy lekerekített sarkú *rect* elem leírása *path* elemekkel az alábbi lenne:

```
Moveto(x+rx,y), Lineto(x+width-rx,y), Arc(x+width,y+ry) large-
arc="0" sweep-flag="1", V(x+width,y+height-ry), Arc(x+width-
rx,y+height), H(x+rx,y+height), A(x,y+height-ry), V(x,y+ry),
Arc(x+rx,y)
```

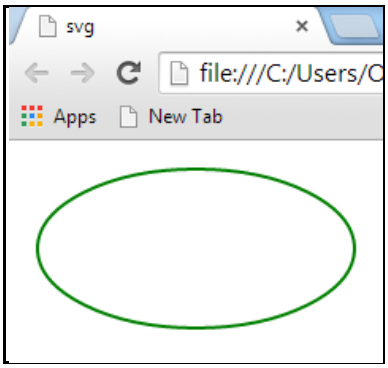
2.) A *circle* **kör** elem a középpontjának (*cx, cy*) koordinátáival és a (r) sugarával adható meg. (A körvonal a „3 órától” kiindulva az óramutató járásával megegyező irányban halad - ennek nem itt, hanem máshol lesz jelentősége.)

```
<!doctype html>
<html lang="hu">
  <svg width="150" height="150">
     <circle cx="80" cy="80" r="60" fill="none"
             stroke="blue" />
  </svg>
</html>
```



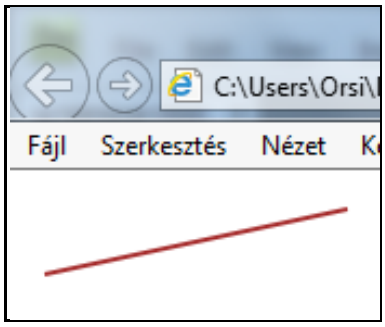
3.) Az *ellipse* **ellipszis** elem a középpontjának (*cx, cy*) koordinátáival, a két sugarával (*rx, ry*) és a tengelyeinek irányával adható meg. Jelen kódolásban a tengelyek egybeesnek a (*x, y*) koordináta-tengelyekkel, az ettől eltérő helyzet egyéb, később ismertetett technikával valósítható meg. Az *rx=ry* eset kört eredményez. (Az ellipszis íve a „3 órától” kiindulva a „9 óra” irányában halad - ennek nem itt, hanem máshol lesz jelentősége.)

```
<!doctype html>
<html lang="hu">
  <svg width="250" height="200">
     <ellipse cx="110" cy="60" rx="100" ry="50" fill="none"
              stroke="green" />
  </svg>
</html>
```



4.) A *line* **egyenes vonal** elem a kezdeti *(x1, y1)* és végpontjának *(x2, y2)* a helyzetével specifikálható. Az aktuális koordinátarendszerhez (x,y tengelyekhez) igazított *x,y* értékek a bal felső saroktól számítandók.

```
<!doctype html>
<html lang="hu">
  <svg width="200" height="50">
     <line x1="10" y1="40" x2="150" y2="10" stroke="brown"
           fill="none" />
  </svg>
</html>
```

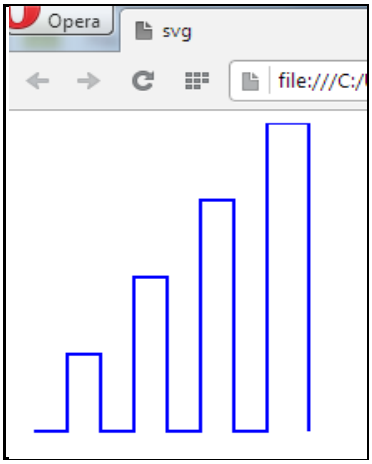


Megjegyzés: Egy *line* egyenes vonal elem leírása *path* kódolással az alábbi lenne:

Moveto(x1,y1***), Lineto(***x2,y2***)***

5.) A *polyline* összekötött egyenes vonalszakaszokból álló elem tipikusan nyílt alakzatokat ír le, melyet a vonalszakaszok találkozási pontjai koordinátáinak (*xi,yi*) vessző nélküli, egy szóközös felsorolása (*points=”…”)* definiál. (Ha páratlan számú koordináta-értéket adunk meg, akkor hibás a kódolás, valamelyik pont egyik koordinátaértéke hiányzik.)

```
<!doctype html>
<html lang="hu">
  <svg width="200" height="250">
    <polyline fill="none" stroke="blue" points="10,200
              30,200 30,150 50,150 50,200 70,200 70,100 90,100
              90,200 110,200 110,50 130,50 130,200 150,200
              150,0 175,0 175,200" />
  </svg>
</html>
```

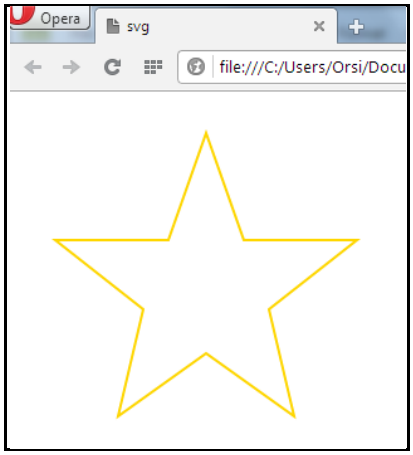


Megjegyzés: Egy *polyline* elem leírása *path* kódolással az alábbi lenne:

Moveto(x1,y1***), Lineto(***x2,y2***), Lineto(***x3,y3***)……Lineto(***x17,y17***)***

6.) A *polygon* elem összekötött egyenes vonalszakaszokból álló zárt alakzatokat ír le, melyet a vonalszakaszok találkozási pontjai koordinátáinak (*xi*,*yi*) vessző nélküli, egy szóközös felsorolása (*points*=”…”) definiál. (Ha páratlan számú koordináta-értéket adunk meg, akkor hibás a kódolás, valamelyik pont egyik koordinátaértéke hiányzik.)

```
<!doctype html>
<html lang="hu">
  <svg width="300" height="250">
    <polygon fill="none" stroke="gold" points="150,25 180,110
              270,110 200,165 220,250 150,200 80,250 100,165
              30,110 120,110" />
  </svg>
</html>
```



Megjegyzés: Egy *polygon* elem leírása *path* kódolással az alábbi lenne:

Moveto(x1,y1***), Lineto(***x2,y2***),……Lineto(***x10,y10***), Z***

Az alakzatok kódolásához felhasznált utasítások:

M = moveto	kiindulási pont
L,l = lineto	egyenes vonal
H,h = horizontal lineto	vízszintes vonal
V,v = vertical lineto	függőleges vonal
C,c = curveto (cubic Bézier curve)	négyzetes Bezier-görbe
Q,q = quadratic Bézier curve	kvadratikus Bezier-görbe
S, s	folyamatosan illeszkedő négyzetes Bezier-g.
T, t	folyamatosan illeszkedő négyzetes Bezier-g.
A,a = elliptical arc	elliptikus ív
Z = closepath	alakzat bezárása

4.2. Szöveg

Szöveget a *<text>……<\/text>* páros címkével lehet létrehozni. Tehát a „Legyen szép napod!” mondat grafikus szövegként:

```
<svg>
   <text>
       Legyen szép napod!
   </text>
</svg
```

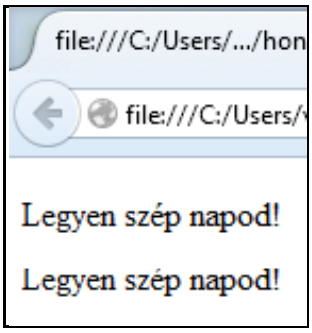
A szöveget csak légypiszokként lehet így a legtöbb böngészőben látni. A formázás tárgyalását megelőzve egy dolgot ismét előre kell venni: a szöveg kezdőpontjának koordinátáit *x=”….”* és *y=”…..”* módon lehet kódolni, és az alapértelmezett *0,0* kezdőpont a legtöbb böngészőben csak a betűk alját teszi láthatóvá. Ezért kezdettől adjuk meg az *x,y* koordinátákat:

```
<!doctype html>
<html lang="hu">
   <svg width="200" height="100">
      <text x="10" y="10">
       Legyen szép napod!
      </text>
   </svg>
</html>
```

Egy dokumentumban lehetnek vegyesen HTML és SVG szövegek is, pl.:

```
<!doctype html>
<html lang="hu">
   <p>Legyen szép napod!</p>
   <svg width="200" height="100">
      <text x="0" y="10">Legyen szép napod!</text>
   </svg>
</html>
```

Alapértelmezett megjelenítésükben nincsen különbség:



A *fill* (kitöltés) és *stroke* (kontúrvonal színe) jellemzők az alakzatokból ismert módon és a CSS-ből ismert színértékekkel működnek.

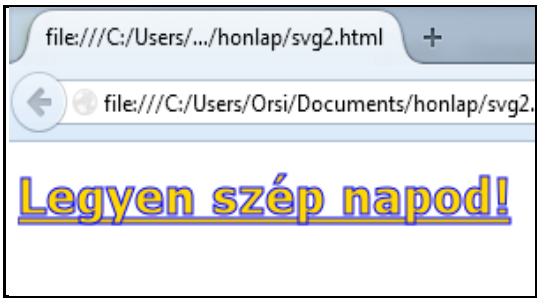
A grafikus szöveg (beleértve a betűtípusokat is) formázásának jelentős része a CSS-nél tárgyaltakkal analóg módon történik:

betűcsalád (*font-family*):
- gyűjtő családnév (*generic family keyword*): serif, sans-serif, monospace, cursive, fantasy
- specifikus családnév (*font-family name*): nagyon sok

betűvastagság (*font-weight*): normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900
betűstílus (*font-style*): normal, italic, oblique
betűváltozat (*font-variant*): normal, small-caps
betűméret (*font-size*): méret
betű kiterjedés (*font-stretch*): normal, wider, narrower, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded
betűméret igazítás (*font-size-adjust*): méret
szöveg díszítése (*text-decoration*): none, underline, overline, line-through
sortörés (*line breaking*): line break, word break, hyphens, word-wrap, overflow-wrap
igazítás (*alignment*): text-align, text-align-last
soron belüli betűköz, szóköz (*spacing within lines*): letter-spacing, word-spacing (méret)
sorköz (*line-spacing*): line-height (itt megegyezik a *font-size*-al)
szövegdoboz (*text-layout, auto-wrapped text*): width, height
szövegbehúzás (*edge effect*): text-indent
elválasztó karakterek kezelése (*white space handling*): white-space (normal, pre, nowrap, pre-wrap, pre-line)

Néhány formázási jellemzőt/értéket rendelve a korábbi mondathoz:

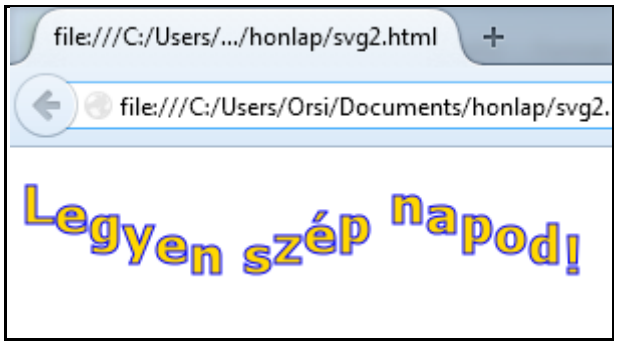
```
<!doctype html>
<html lang="hu">
   <svg width="400" height="100">
     <text x="0" y="30" font-family="Verdana" font-size="25px"
           font-weight="bold" fill="gold" stroke="blue" word-
           spacing="30px" text-decoration="underline">
        Legyen szép napod!
     </text>
   </svg>
</html>
```



X,y **koordinátaértékek listája** is megadható, ekkor az első értékhez az első betű, a második értékhez a második betű, stb. rendelődik. Ha az értékek száma meghaladja a betűk számát, akkor a listában a betűk számát követő értékek nem lesznek figyelembe véve. Ha az értékek száma kevesebb a betűk számánál, akkor a szülőelem/eredeti érték vonatkozik a további betűkre is.

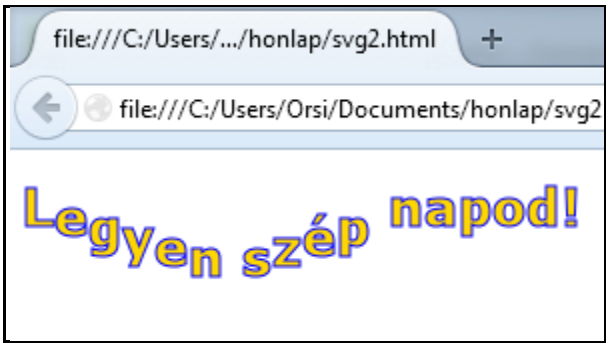
Például ha az értékek száma meghaladja a betűk számát (az aláhúzást kivettük, mert itt zavaró lenne):

```
<!doctype html>
<html lang="hu">
   <svg width="350" height="100">
     <text x="0" y="30 35 40 45 50 55 60 55 50 45 40 35 30 35
           40 45 50 55 60 55 50" font-family="Verdana" font-
           size="25px" font-weight="bold" fill="gold" stroke=
          "blue" word-spacing="30px" text-decoration="none">
        Legyen szép napod!
     </text>
   </svg>
</html>
```



Ha az értékek száma kevesebb a betűk számánál:

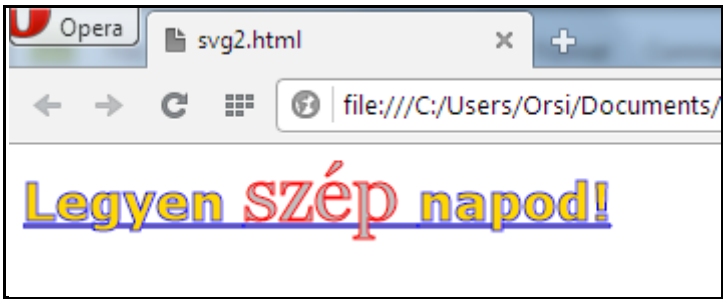
```
<!doctype html>
<html lang="hu">
   <svg width="350" height="100">
     <text x="0" y="30 35 40 45 50 55 60 55 50 45 40 35 30"
           font-family="Verdana" font-size="25px" font-weight=
          "bold" fill="gold" stroke="blue" word-spacing="30px"
           text-decoration="none">
        Legyen szép napod!
     </text>
   </svg>
</html>
```



A *<tspan>……..</tspan>* páros címkével kiválaszthatók szövegrészek formázás céljából. A kijelölt részekhez a *<text>……</text>* -el megegyező módon kódolhatók a jellemzők/értékek.

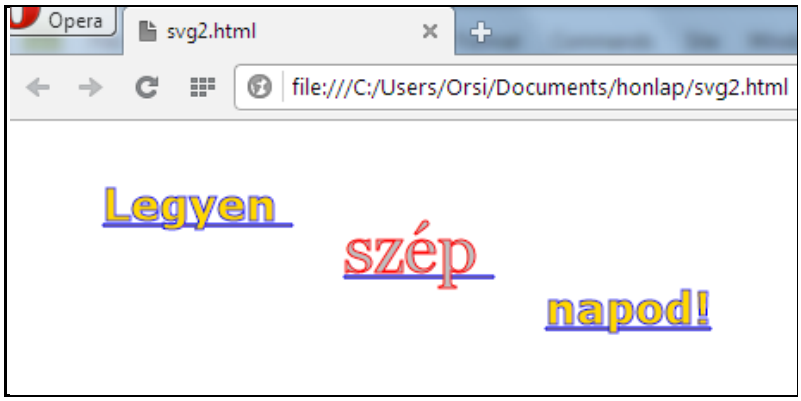
Például a fenti szövegből a „szép” szó formázását megváltoztatva:

```
<!doctype html>
<html lang="hu">
   <svg width="350" height="100">
     <text x="0" y="30" font-family="Verdana" font-size="25px"
            font-weight="bold" fill="gold" stroke="blue" word-
            spacing="30px" text-decoration="underline">
        Legyen <tspan font-family="Georgia" font-size="40"
               font-weight="normal" fill="silver" stroke="red"
               >szép </tspan>napod!
     </text>
   </svg>
</html>
```



Szöveg formázására további jellemző a ***dx*** és ***dy***, melyekkel a szöveg relatív helyzete (a koordinátatengelyek mentén való eltolása) változtatható meg. A már kijelölt és formázott „szép” szóra és a „napod” szóra alkalmazva a *dx* és *dy* jellemzőket:

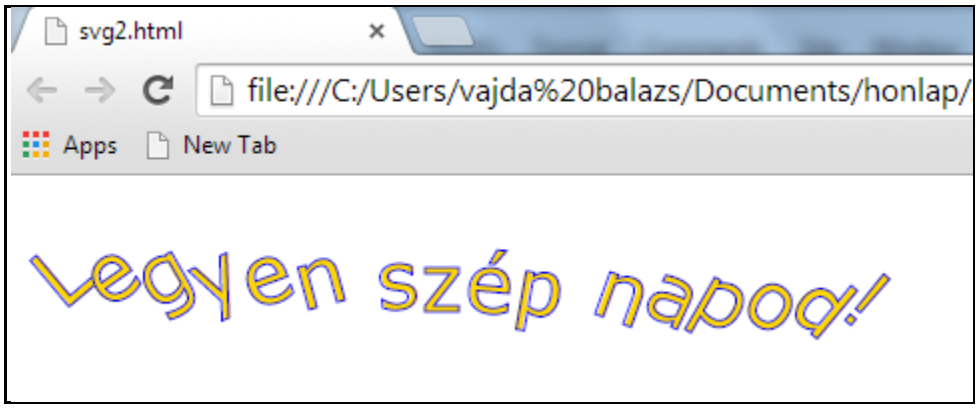
```
<!doctype html>
<html lang="hu">
   <svg width="500" height="300">
     <text x="50" y="50" font-family="Verdana" font-size="25"
            font-weight="bold" fill="gold" stroke="blue" word-
            spacing="30px" text-decoration="underline">
        Legyen <tspan dx="30" dy="30" font-family="Georgia"
            font-size="40" font-weight="normal" fill="silver"
            stroke="red">szép </tspan><tspan dx="30" dy="30">
            napod!</tspan>
     </text>
   </svg>
</html>
```



Ugyancsak szövegformázást tesz lehetővé a ***rotate*** jellemző, melynek értéke(i) a vízszintessel bezárt szög(ek). Ha értékek listáját adjuk meg, akkor a betűnkénti hozzárendelés szabálya megegyezik az *x*,*y* listáknál látottakkal.

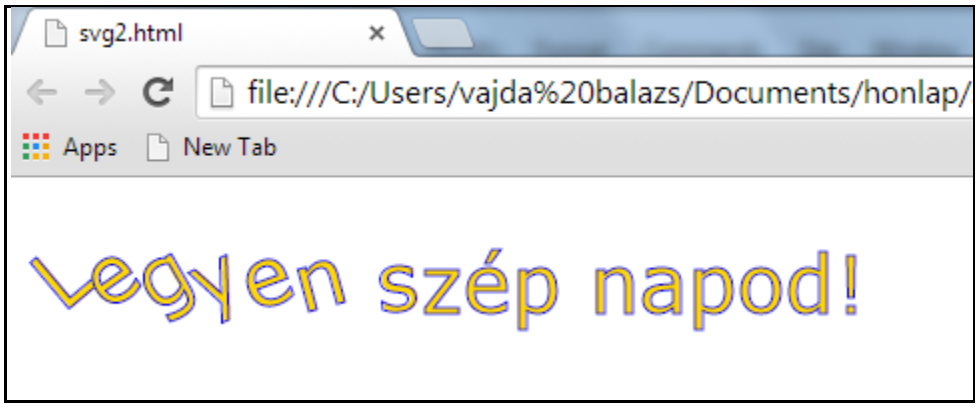
Például ha az értékek száma meghaladja a betűk számát (az aláhúzást kivettük, mert itt zavaró lenne):

```
<!doctype html>
<html lang="hu">
   <svg width="500" height="300">
      <text font-family="Verdana" font-size="40" x="20" y="60"
            rotate="-40,-35,-30,-25,-20,-15,-10,5,0,5,10,15,
            20,25,30,35,40" fill="gold" stroke="blue">
        Legyen szép napod!
      </text>
   </svg>
</html>
```



Ha az értékek száma kevesebb a betűk számánál:

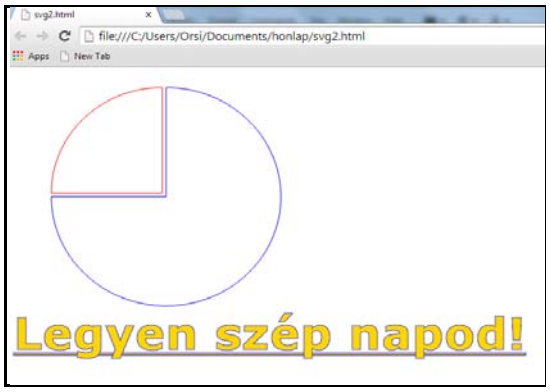
```
<!doctype html>
<html lang="hu">
   <svg width="500" height="300">
      <text font-family="Verdana" font-size="40" x="20" y="60"
         rotate="-40,-35,-30,-25,-20,-15,-10,-5,0" fill="gold"
         stroke="blue">
       Legyen szép napod!
      </text>
   </svg>
</html>
```

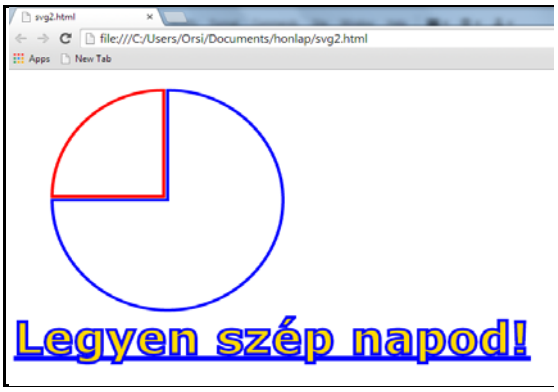


Tekintettel arra, hogy a szöveget is grafikaként kezeli az SVG, vannak az alakzatoknak és szövegeknek közös formázási jellemzői/értékei. Ilyenek pl. a vonalvastagság:

stroke-width=”…….”

```
<!doctype html>
<html lang="hu">
   <svg width="800" height="500">
      <path d="M200,170 h-150 a150,150 0 1,0 150,-150 Z"
            stroke="blue" fill="none" stroke-width="4" />
      <path d="M195,165 v-145 a145,145 0 0,0 -145,145 Z"
            stroke="red" fill="none" stroke-width="4" />
      <text x="0" y="380" font-family="Verdana" font-size="60"
            font-weight="bold" fill="gold" stroke="blue" word-
            spacing="30px" text-decoration="underline" stroke-
            width="4">
         Legyen szép napod!
      </text>
   </svg>
</html>
```



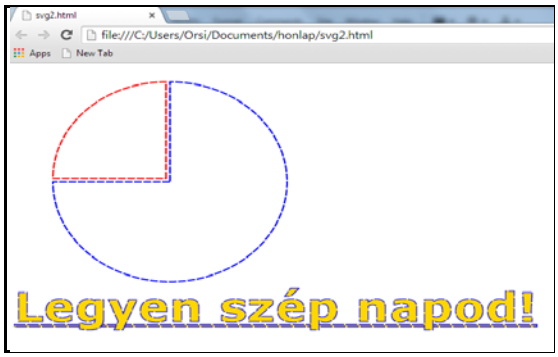


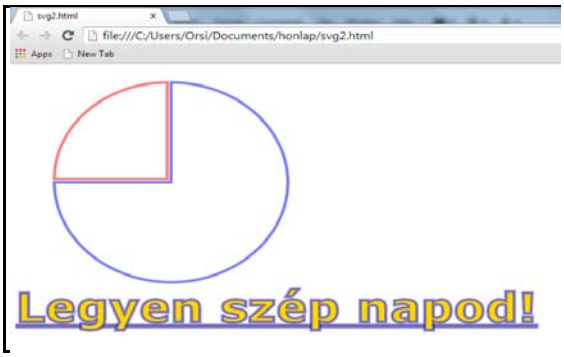
alapértelmezetten (bal oldalon) *1px* a kódolt vonalvastagság, jobb oldalon *4px*.

Szaggatott vonal váltakozó szakaszainak hossza (az értékek vesszővel elválasztva) és a vonalszínek átlátszósága (0 – között, lásd színeknél):

stroke-dasharray=”10,2***”*** (balra, *10px* és *2px*)

stroke-opacity=”0.5***”*** (jobbra, 0.5 értékkel)





A *stroke-opacity*-vel analóg módon definiálható a kitöltőszín átlátszósága, a ***fill-opacity***.

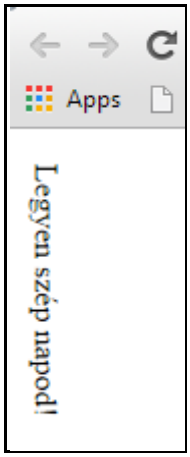
SVG-kódolással (díszítésként, figyelemfelhívásként vagy design-elemként) - a Firefox kivételével - valamennyi böngésző megjeleníti a vízszintes, balról jobbra haladó szöveget függőleges szövegként is.

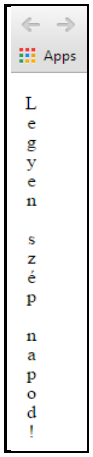
A függőleges, fentről lefelé haladó szöveggé alakítás a *writing-mode* (írásmód) tulajdonság *tb* (top-to-bottom) értékével valósítható meg:

```
<!doctype html>
<html lang="hu">
   <svg width="200" height="100">
      <text x="10" y="10" writing-mode="tb">
        Legyen szép napod!
      </text>
   </svg>
</html>
```

A *writing-mode* tulajdonságot a *glyph-orientation-vertical* tulajdonság *0* értékével kiegészítve a függőleges szöveg karakterei vízszintes irányba fordíthatók:

```
<!doctype html>
<html lang="hu">
   <svg width="200" height="100">
      <text x="10" y="10" writing-mode="tb" glyph-orientation-
            vertical="0">
        Legyen szép napod!
      </text>
   </svg>
</html>
```





Megjegyzések:

- a vízszintesből függőlegesbe forgatás következtében az *svg* elem magasságát ennek megfelelően növelni kell
- amennyiben a vízszintes szöveg formázásában szerepet játszott az *y* koordináták változtatása, azt a függőlegesbe forgatás után újra kell írni: enélkül egymásra torlódhatnak a karakterek és olvashatatlanná válhat a szöveg

4.3. Szöveg illesztése alakzatokhoz

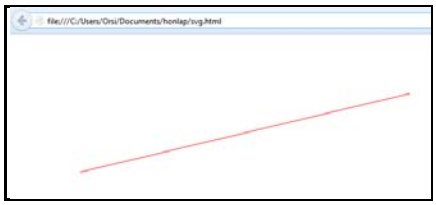
A függőleges írásmódhoz hasonló design-elem a szövegnek valamilyen alakzathoz való illesztése.

Kódolása két részből áll:

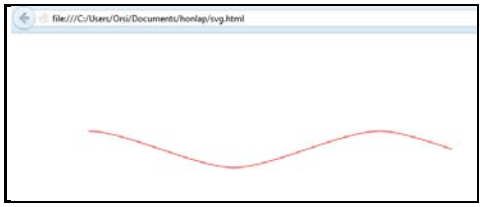
a) *<defs>…</defs>* (defs=definition) páros címkék között definiáljuk azt az alakzatot, melyhez a szöveget illeszteni kívánjuk

b) a *<text>…..</text>* kódrészben *<textPath>……..</textPath>* páros címkék között hivatkozunk a definiált alakzatra (*xlink:href=”#.....”*)

Két példán, egy egyenes és egy görbe vonalra illesztésen mutatjuk be a fentieket. Az alakzatok az alábbiak lesznek:



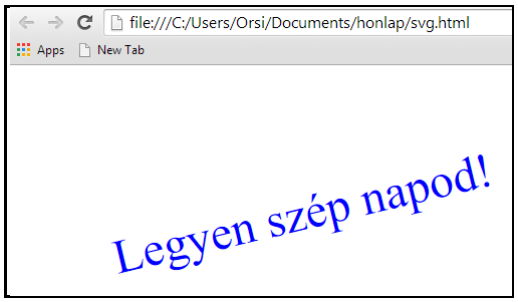
d="M100 200 L900 10"
(felfelé ferde egyenes)



d="M100 125 C150 125 250 175 300 175 C350 175 450 125 500 125 C550 125 650 175 700 175 C750 175 850 125 900 125"(négyzetes Bezier-görbékből alkotott hullámvonal)

A ferde egyenesre illesztés kódja és megjelenítése:

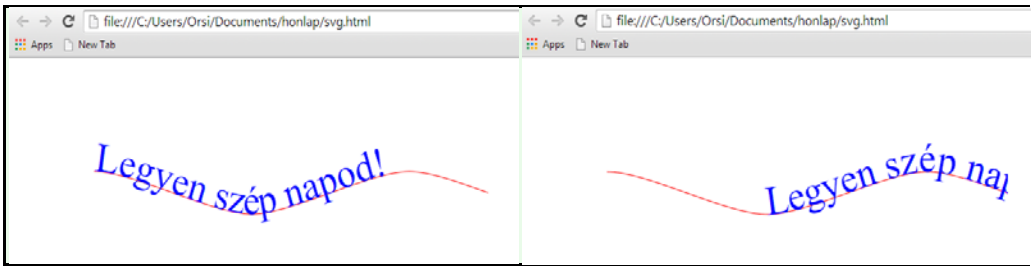
```
<!doctype html>
<html lang="hu">
  <svg width="600" height="300">
    <defs>
      <path id="egyenes" d="M100 200 L900 10" fill="none" stroke="black" />
    </defs>
    <text font-size="48" fill="blue">
    <textPath xlink:href="#egyenes">
      Legyen szép napod!
    </textPath>
    </text>
  </svg>
</html>
```



A ferde egyenesre illesztés kódja és megjelenítése:

```
<!doctype html>
<html lang="hu">
  <svg width="600" height="300">
    <defs>
      <path id="gorbe" d="M100 125 C150 125 250 175 300 175
             C350 175 450 125 500 125 C550 125 650 175 700 175
             C750 175 850 125 900 125" />
    </defs>
    <text font-size="48" fill="blue">
    <textPath xlink:href="#gorbe">
      Legyen szép napod!
    </textPath>
    </text>
  </svg>
</html>
```

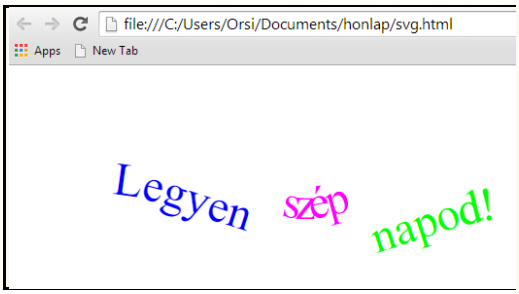
A szöveg kezdetének helyzete a *startOffset* jellemző értékével specifikálható. Az (alapértelmezett) *0%* az alakzat kezdetét jelenti, a *100%* az alakzat végpontja (ekkor semmi nem látszik a szövegből, hiszen véget ér az alakzat, mikor a szöveg kezdődne). Ha a kezdőpont eltolása következtében a szöveg nem fér ki teljes hosszában az alakzatra, a túlnyúló rész nem kerül megjelenítésre:



<textPath xlink:href="#gorbe" startOffset="0%"> *<textPath xlink:href="#gorbe" startOffset="25%">*

Az alakzathoz viszonyított szöveghelyzet vagy a szöveg egyéb tulajdonsága a *<tspan>……</tspan>* elemekkel tovább formázható. Például a *szép* és *napod* szavak relatív pozíciójának és színének a megváltoztatásával a kódolás és a megjelenítés:

```
<text font-size="48" fill="blue">
  <textPath xlink:href="#gorbe">
    Legyen <tspan dx="20" dy="-25" fill="fuchsia">szép</tspan>
    <tspan dy="50" fill="lime"> napod!</tspan>
  </textPath>
</text>
```



Figyelem! SVG grafikára alkalmazhatók továbbá gradiensek, (szín)szűrők, árnyékok, markerek, transzformációk, animációk, stb.

SVG 2 FOLYT. KÖV.

5. FÜGGELÉK - Történeti összefoglaló

5.1. Előzmények I. (Internet)

Az Internet létrejötte az infokommunikációs technika fejlődése során szükségszerű volt. Megvalósulásának idejét és konkrét módját azonban végül a világpolitika is alakította.

1957 Nemzetközi Geofizikai Év volt – ennek alkalmából már korábban nyilvánosan tervbe vette az USA az első mesterséges égitest, az „Explorer 1" műhold felbocsátását, ez azonban több sikertelen kísérletet követően csak 1958 január végén valósult meg. A Szovjetunió viszont titkos felkészülés után váratlanul, 1957 október 4-én földkörüli pályára juttatta a „Szputnyik 1"-et, mely 3 hónapnyi sikeres tevékenység után, küldetését még az „Explorer 1" felbocsátása előtt befejezve, 1958 január 4-én elégett a légkörben.

A „Szputnyik 1" sikere nemcsak az akkori két szuperhatalom űrversenyének kezdetét jelentette, hanem egyben sokkolta az amerikai kormányzatot és tudományos életet is, mert a II. világháború után biztosnak hitt technológiai fölényüket látták veszélyeztetve. Az USA kormányzata 1958-ban létrehozta az ARPA-t (Advanced Research Projects Agency = Fejlett Kutatási Projektek Ügynöksége), melynek keretében a következő évtizedben hatalmas összegeket irányítottak a stratégiai technológiák kutatásába, oktatásába és katonai/polgári bevezetésükbe.

Az átfogó, nemzetbiztonsági jelentőségű fejlesztések egyik kiemelt területe a decentralizált, hibatűrő, sebezhetetlen, országos katonai adatátviteli hálózat létrehozása volt. A kutatások hatékonyságának fokozására szintén hasznos eszköz lett az egyes kutatóhelyek számítógépeinek adatátviteli hálózatokkal történő összekapcsolása. (A szovjetek központilag tervezett és irányított kutatásaival szemben az amerikai decentralizált, független kutatóhelyeken ugyanis gyakran dolgoztak hasonló témákon párhuzamosan anélkül, hogy tudtak volna egymás eredményeiről, így az újdonságokat – állítólag - átlagosan másfélszer kellett feltalálniuk.)

A hálózati fejlesztések fontos állomása volt 1965-ben a csomagkapcsolt átviteli protokoll (X.25) kifejlesztése, mely az adatok csomagokra bontásával és többutas átvitelével egy hálózat adatátviteli kapacitásának és az adatátvitel biztonságának növelését tette lehetővé.

1969 végén kezdett működni kutatóhelyek számítógéprendszereinek összekapcsolásával az alapvetően katonai célokat szolgáló, a kormányzati ügynökség nevét viselő ARPANET, mely 1971-re már 15 intézetet kötött össze. 1974-ban publikálták az TCP/IP csomagkapcsolt protokoll-családot (Transmission Control Protocol / Internet Protocol), mely a mai napig alapvető jelentőségű, és amelyben először használták az „Internet" elnevezést. Ez az angol „internetwork" azaz „hálózatok közötti" kifejezésből származik, és azóta az egyik legismertebb nemzetközi szóvá vált a világon.

1983-ban leválasztották az addig szigorúan őrzött ARPANET-ről MILNET (Military Network) néven a szorosan vett katonai alkalmazásokat, az ARPANET-et pedig a TCP/IP protokoll-családra állították át - ezzel megszületett a mai fogalmaink szerinti, általánosabb felhasználhatóságú Internet (ezt eredetileg nagybetűvel, tulajdonnévként használták).

1985-től a National Science Foundation (Amerikai Tudományos Alapítvány) épített ki NSFNET néven szintén TCP/IP alapú hálózatot, melyet 1988-ban az ARPANET-hez kapcsoltak és megnyitották a kereskedelmi alkalmazások előtt is. Az ARPANET formálisan 1989-ben szűnt meg, átadva helyét a fejlettebb NSFNET és a kialakuló magáncéges gerinc-

hálózatoknak. Így létrejött a tudományos és üzleti, kormányzati és magán, helyileg és globálisan összekapcsolt számítógéphálózatok átfogó rendszere, mely központi kormányzás nélküli hardver és szoftver infrastruktúrán alapul, és a csatlakozott számítógépek közötti kapcsolat révén univerzális adatátvitelt nyújt.

5.2. Előzmények II. (hiperszöveg, jelölőnyelv)

A hiperszöveg (angolul „hypertext”) olyan szöveg, melyben nem csak lineárisan - tehát a nyelvek többségénél folyamatosan balról jobbra, néhány nyelv esetében jobbról balra vagy fentről lefelé - lehet haladni, hanem kapcsokat, hivatkozásokat (angolul „hyperlink”-eket) is tartalmaz, melyek más szövegekhez vagy szövegrészekhez vezetnek. A hiperszöveg tehát egy nemlineáris, sokközpontú, digitális közegben hálószerűen (angolul a háló =”web”) felépülő szövegrendszer, melynek elemeit hiperhivatkozások kötik össze. A „hiper” kifejezés az írott szöveg régi, lineáris korlátainak meghaladását jelenti (szótárakban, enciklopédiákban, lexikonokban nyíl vagy index formájában tett hivatkozásokkal már régi törekvés volt ez).

A hiperszöveg egy koncepció, melynek több megjelenési formája van. Modellezését már régóta kutatták, maga a hiperszöveg kifejezés 1965-ből T. Nelson amerikai számítógép kutatótól származik. Ő használta a hipermedia kifejezést is, mikor a hiperszöveg nem csak szöveget, hanem hangot, képet, klip-et, vagy grafikát is tartalmaz. Bár ma ez a gyakoribb alkalmazás, maga a hipermedia kifejezés mégsem terjedt el.

Az 1970-es években az IBM-nél C. Goldfarb, E. Mosher és R. Lone fejlesztették ki a nevük kezdőbetűjéről GML-nek, majd hivatalos névvé változtatva (Generalized Markup Language = általános jelölőnyelv) nevezett jelölőnyelvet nagygépes dokumentumkezelésre. 1986-ban ISO szabvány lett SGML (Standard Generalized Markup Language = Szabványos Általános Jelölőnyelv) néven, ezt tekintik a hiperszövegek világában a „jelölőnyelvek atyjának”.

Sokféle jelölőnyelv (néha leírónyelvnek is fordítják magyarra) létezik, legtöbbjük a szövegbe helyezett jelölő utasításoknak (angolul „tag=címke”) a hiperszöveg adatfolyamába vagy fájljába való elhelyezésével működik. Az általános jelölőnyelvben a jelölő utasításoknak csak a nyelvtana van megadva, a kódolásra használt építőkockák („tag”-ek) jelentésének értelmét az adott alkalmazás szabja meg. Az SGML legismertebb és legelterjedtebb alkalmazásában, a világhálót uraló HTML-ben (Hypertext Markup Language = hiperszöveg jelölőnyelv) a jelölő címkékhez konkrét jelentést és véges értékkészletet rendeltek.

5.3. Előzmények III. (HTTP, HTML, www)

A világháló koncepciója a svájci-francia határon lévő CERN-ben (Centre Européen pour la Recherche Nucléaire = Nukleáris Kutatások Európai Központja), Tim Berners-Lee angol fizikus ötletéből született. 1989 márciusában javaslatot tett egy az Internet-en alapuló információs rendszer kiépítésére, mely az intézet valamennyi tudományos eredményének publikálására és az azokhoz való azonnali és közvetlen hozzáférésre irányult egy módosított jelölőnyelv és erőforrás-azonosítók együttes használatával. Bár javaslatát akkor elutasították, a CERN 2009 márciusában emlékezett meg a világháló születésének 20. évfordulójáról, és a W3C idén márciusban ünnepelte a 25. évfordulót. Az átdolgozott javaslatot végül 1990 októberében fogadták el, az első működő demonstrációt 1990 végén mutatták be, a működő rendszert a CERN számítógéprendszerén 1991 májusában helyezték üzembe. A világ első web-szerverét pedig a CERN - az akkori Internet egyik legnagyobb európai csomópontja -

egy mikroszámítógépén 1991 augusztusában indították el (tehát bármelyik időpont lehetne a világháló születésnapja).

Az első honlap címe *http://info.cern.ch/hypertext/WWW/TheProject.html* volt, melyen a HTTP protokoll és HTML jelölőnyelv specifikációját bocsátották nyilvános vitára és a technológia propagálására. A cél egy globális hipertér (az Interneten a hiperszövegek világa) létrehozása volt, melyen az Internethez kapcsolt bármely számítógépről felvihetők, kereshetők és elérhetők információk. Ezt tükrözi az ekkor megalkotott világháló (www=worldwide web) elnevezés is. A világháló azonban nem azonos az Internettel, hanem annak egy alkalmazása, felhasználva az Internet nyújtotta lehetőségeket.

A HTML-nyelvet az SGML egy célirányos alkalmazásaként fejlesztették ki, a HTTP (Hypertext Transfer Protocol = Hiperszöveg Átviteli Protokoll) pedig a TCP/IP feletti alkalmazás-protokoll, melyet a böngészők ma is használnak a weboldalak elérésére (lásd *http://www.....* kezdetű honlap címek).

1993-ban a CERN-ben döntés született az addig elért eredmények ingyenes közkinccsé tételéről. A további fejlesztések és egyeztetések az amerikai IETF-hez (Internet Engineering Task Force) kerültek, mely nem sokkal előtte vált amerikai kormányszervből nyitott, tudományos szervezetté.

5.4. Előzmények IV. (W3C, CSS, PNG, SVG)

Az IETF egy általában negyedévenként ülésező, jogi személyiséggel és főállású munkatársakkal nem rendelkező, döntően amerikai súlyú, internet szabványokkal foglalkozó informális szervezet, ezért felmerült az igény egy állandó nemzetközi intézmény alakítására a világháló folyamatos fejlesztésének és szabványosításának elősegítésére.

1994 októberében hozták létre Tim Berners-Lee vezetésével és az Európai Bizottság támogatásával a World Wide Web Consortium-ot (általánosan használt rövidítése „W3C”) egy-egy amerikai és európai központtal az MIT-nél (Massachusetts Institute of Technology) és a CERN-nél. 1994 decemberében azonban a CERN-nél eldöntötték az LHC (Nagy Hadron Ütköztető) megvalósítását, mely minden jövendőbeli anyagi forrásukat igénybe vette, és így nem tudták a továbbiakban vállalni a világhálós fejlesztéseket. Helyükre az INRIA (Institut National de Recherche en Informatique = Nemzeti Informatikai Kutató Intézet) lépett, 1996-ban pedig a japán Keio Egyetem csatlakozott ázsiai központként a W3C-hez. 2003-ban az INRIA helyét az ERCIM (European Research Consortium in Informatics and Mathematics = Európai Informatikai és Matematikai Kutató Konzorcium) váltotta fel, majd a kínai Beihang egyetemet nevezték ki másik ázsiai központtá. A világ sok országában nemzeti irodák alakultak, melyeken keresztül be lehet kapcsolódni a standardok kidolgozásába. A W3C Magyar Iroda az MTA SZTAKI keretein belül első kelet-közép-európai irodaként jött létre még 1995-ben.

Az IETF és W3C között a hatáskörök tisztázása és átadása néhány évig tartott. Még az IETF keretében dolgozták ki 1995-ben a HTML 2.0 verziót, mely az első általánosan elfogadott standard lett. A HTML 3.0-át szintén az IETF kezdte el, de már nem fejezte be, a HTML munkacsoportjukat feloszlatták, a munkát a W3C vette át. A HTTP protokoll fejlesztése viszont egy kerülő után végül is az IETF-nél maradt.

A W3C által 1997-ben először publikált HTML 3.2 verzió csak néhány hónapig élt, viszont az 1997 végén életbe lépett HTML 4.0 és a hozzá kifejlesztett CSS stíluslapnyelv ill. PNG képformátum megalkotása stabilitást hozott a weblapkészítési szabványok terén.

A W3C javaslatkérésére 9 stíluslapnyelvre érkezett ajánlat, ebből kiválasztottak kettőt munkapéldánynak, majd ezekből alakították ki a CSS1-et 1996 végén. A javított és bővített CSS2 verziót 1998 elején fogadták el (ennek letisztázása és véglegezése a jelenleg hatályos CSS2.1 és a modulonként továbbfejlesztés alatt álló CSS3).

A PNG („Portable Network Graphics" = hordozható hálózati grafika) képformátumot a W3C a világhálós alkalmazások ingyenességére törekedve fejlesztette ki. Veszteségmentes képtömörítésre az Unisys/Compuserve-nek a GIF formátumra szabadalmi védettsége volt, a PNG-t (tréfásan úgy is értelmezhető a rövidítés, hogy „**P**NG is **n**ot **G**IF" = a PNG nem GIF) 1996 végén publikálták, és azóta a weblapok fontos alkotóelemévé vált.

1998 elejére tehát a W3C kézbe vette és megalapozta a világháló fontos standardjait - melyek valójában „csak" technikák és ajánlások, hiszen nincsen joga szabványokat alkotni. Bár ezek mindenki számára ingyenesen elérhetők és könnyen használhatók voltak, elfogadtatásuk és elterjesztésük volt a kulcskérdés a „web" robbanásszerű fejlődéséhez.

Fontos szerepet játszott a W3C ajánlások használatának ösztönzésében a profi webfejlesztők által 1998-ban alapított Web Standards Project. A költségek és komplexitás csökkentésére és a hosszútávú megjeleníthetőségre hivatkozva 2001-re meggyőzték a Microsoft-ot és Netscape-et (az akkori két vezető böngészőgyártót), hogy támogassák a HTML és CSS ajánlásokat, fejlesztéseiket ebben az irányban folytassák - e nélkül a web (böngészőnkénti) inkompatibilis tartalmakra szakadhatott volna szét. (2002-től az alapítók szétszóródtak, azóta a szervezet főleg oktatással foglalkozik. A Netscape időközben csődbe ment, az Internet Explorer viszont az IE6 – IE7 – IE8 fejlesztésekkel lassan, de folyamatosan hozzáidomult az évtized végére a HTML 4.0 / 4.01 ill. CSS2.0 / 2.1 ajánlásokhoz.)

Az SVG vektorgrafikai jelölőnyelv – a CSS-hez hasonlóan – a W3C „zöldmezős" fejlesztése, 1.0 verziója 1998-2001 között készült, 1.1 verzióját 2003-ban fogadták el. Mivel az Internet Explorer 6-7-8 nem támogatták, használhatósága évekig korlátozott maradt. Az IE9 – csakúgy mint a HTML5 és CSS3 terén – az SVG széleskörű alkalmazhatóságában is áttörést jelentett, a fejlesztése megint megindult. 2011-ben elfogadták az 1.1 második, javított kiadását, és 2012-től zajlik a 2.0 kidolgozása.

5.5. Előzmények V. (XHTML)

A világháló további fejlődésében a W3C szerepét és súlyát néhány alapvető tényező részben korlátozza, részben erősíti. Valamennyi szereplőnek nyilvánvalóan közös érdeke az egységes standardok használata, ugyanakkor korlátozó tényező, hogy

- a konzorcium nem fogadhat el kötelező érvényű szabványokat, csak ajánlásokat tehet
- a „web" gazdasági jelentősége annyira megnőtt, hogy nagy anyagi érdekeket mozgósít, az érintett cégek elemi érdeke a fejlődés irányának befolyásolása és a piacszerzés
- a webhelyek számának roppant mértékű növekedése csak olyan fejlesztésekre ad reális alapot, melyek megőrzik a visszafelé kompatibilitást, és a változtatások elfogadtathatók a hatalmas számú, független, heterogén, és esetenként ellenérdekelt szereplőkkel.

A W3C tudományosan indokolt HTML és CSS fejlesztési elképzelései 1998 után egy bő évtizeden keresztül a gyakorlatban kevés elmozdulást eredményeztek:

a) Egyrészt a böngésző programok a CSS ajánlásokat csak lassan és részlegesen kezdték támogatni, és a támogatottság mértékében, ill. bizonyos tulajdonságok értelmezésében továbbra is eltérések voltak. Ezen a W3C két irányban haladva próbált segíteni:

- egyrészt véglegesítette a CSS2.1 verziót, ami tulajdonképpen egy retro-specifikáció (amit amúgy is használnak már a böngészők, az benne van, ami nem terjedt el, azt kihagyták). Ez az ajánlási (*Recommendation*) státuszt 2011. júniusban érte el (az SVG 1.1 második kiadását ugyanezen év augusztusában hagyták jóvá)

- a CSS3-at már modulokra bontva, nem pedig túl nagy falatként a teljes specifikációt egy lépésben publikálva dolgozzák ki. A modulok közül a színekre és kiválasztókra vonatkozó érte el 2011-ben, a média lekérdezések 2012-ben az ajánlási státuszt, a többi modul különböző készültségi fokon van.

b) Másrészt a HTML fejlesztési irányát újra kellett szabni. A HTML 4.0-val a W3C be akarta fejezni a HTML-vonalat, melyet túl lazán felépítettnek és továbbfejlesztésre célszerűtlennek ítélt, és helyette az XHTML-re („Extensible Hypertext Markup Language” = Kiterjeszthető Hiperszöveg Jelölőnyelv), egy XML jelölőnyelven alapuló, szigorúbb és kiterjeszthető szintaktikájú rendszerre tért volna át. (Az XML-t az Open Text Corporation az SGML egy egyszerűsített részhalmazaként fejlesztette ki, és a W3C több más alkalmazásában is erre alapozva kezdett el tovább dolgozni.) A sima átmenet biztosítására 1999 decemberében ill. 2000 januárjában publikálták a HTML 4.01-et és az XHTML 1.0-át, melyeket közel megegyező kódolási szabályokkal definiáltak és az XHTML 1.0 felülről kompatibilis volt a HTML 4.01-el.

Az XHTML továbbfejlesztése (XHTML 1.1) viszont nem volt visszafelé kompatibilis, és – ellentétben a megengedő HTML szabályokkal - már drákói szigorral, a felhasználó felé küldött hibajelzéssel értelmezte a nem szabályszerű kódolást. Ez a gyakorlati felhasználásban megvalósíthatatlannak bizonyult, ugyanis:

Minden böngészőprogram a kezdetektől fogva alkalmazott valamilyen „megengedő” hibakezelési algoritmust, mely a hibás kód értelmezésére irányult, és nem állt le hibajelzéssel (hiszen így a böngésző használhatatlan lett volna). Becslések szerint ugyanis a világhálón a weboldalak száma sok százmilliárd, és ezek 99%-ában (!) van legalább 1 kódolási hiba. Ezekről egyébként a legjobb szándék és felkészültség mellett sem tud feltétlenül a weboldal fejlesztője, hiszen nem tudatosan követi el, és a hibakezelő algoritmus miatt nem kap riasztást a böngészőtől. Hiába tértek át sokan az új, XHTML 1.0 kódolási standardra, a további XHTML verziók (1.1., 2.0) bevezetését a visszafelé kompatibilitás és megengedő hibakezelés megszűnése, a böngészőprogramok gyártóinak elutasítása megakadályozta.

2004 júniusában a W3C tagságában a HTML kontra XHTML vitában szakadás történt - a többség az XHTML 2.0 irányába történő folytatásra, a kisebbség (főleg az akkor alternatívnak nevezett böngészők - Apple, Mozilla, Opera – gyártói) a HTML továbbfejlesztése mellett szavazott. A kisebbségben maradtak létrehozták a WHATWG-t („Web Hypertext Applications Technology Working Group” = Világhálós Hiperszöveges Alkalmazások Technológiáinak Munkacsoportja), egy informális, nemhivatalos, nyitott együttműködési fórumot, melynek célja a HTML 4 visszafelé kompatibilis kiterjesztése volt új alkalmazások és továbbfejlesztett űrlapok támogatására.

Két évig a W3C nem akart tudomást venni a WHATWG-ben folyó munkáról, de a böngészőprogramok XHTML 2.0 támogatásának elmaradása miatt 2006 októberében (a W3C vezetője, és egyben a HTML megalkotója, akkor már Sir Timothy) Berners-Lee elismerte a HTML fokozatos továbbfejlesztésének létjogosultságát, és bejelentette, hogy együttműködik a WHATWG-al, 2007-ben pedig teljes mértékben átvette és W3C tervezetként kezdte kezelni a WHATWG által kidolgozott (Web Forms 2.0 és Web Applications 1.0) tervezeteket, melyeket végül az egyéb HTML fejlesztésekkel együtt HTML5-nek neveztek el.

Újabb két évig a W3C-ben párhuzamosan történtek a HTML5 és XHTML 2.0 fejlesztések, míg 2009 végén az XHTML munkacsoport feloszlott, és 2010 elejétől a HTML5 a W3C egyedüli hivatalos jelöltje a szabványosítási procedúra munkavázlatokkal (WD= working draft) kikövezett útján, melyben 2012 decemberében elérte az ajánlásra jelölti (CR= candidate recommendation) státuszt, egyben 5.1 verziószámmal nekikezdett a továbbfejlesztés irányának felvázolásához. A WHATWG pedig ezután sem oszlatta fel magát, hanem a mai napig tovább folytatja a specifikációkon a munkát „HTML - az élő szabvány", néven, annak eredményeit továbbadva a W3C-nek (és informálisan befolyásolva az irányt és tempót a szabványosításban).

5.6. Rövid böngészőtörténelem

1. (H)őskor

1991-ben Tim Berners-Lee írta az első böngészőt *NeXTstep* platformra *WorldWideWeb* néven. 1992-1993 között jelent meg a *Lynx*, Unix-ra a *Line-mode*, *ViolaWWW*, *Erwise*, *Arena WWW Browser* és *Midas*, MacIntosh-ra a *MacWWW*, Windows-ra a *Cello*, valamint a többplatformos *NCSA Mosaic*, melyet később több cég is licencelt a saját kereskedelmi böngészőjéhez (pl. *Spray Mosaic* és *Spyglass Mosaic* néven), és amelyet a (h)őskor emblematikus böngészőjének, a jövendőbeli világszabványnak tekintettek.

Az Illinois Egyetemen lévő NCSA-ban *(National Center for Supercomputing Applications)* fejlesztették ki az *NCSA Mosaic*-ot, melyet mindjárt Commodore-ra, Windows-ra, Acorn-ra, MacIntosh-ra is portoltak, így a Unix szűk tudományos és műszaki közönségén túl széles kereskedelmi és fogyasztói rétegeket értek el vele.

A *Mosaic* fejlesztőcsapatának vezetője - Marc Andreessen - 1994-ben öt egyetemi kollégájával és diákjával megalapította a Mosaic Communications Corp.-ot, melynek nevét jogi megfontolásból rövidesen Netscape Communications Corp.-ra, az átírt forráskódú böngésző nevét pedig Netscape Navigator-ra változtatták. A világhálóból elsőként profitáló cégként 1995-ben sikeresen tőzsdére mentek, az eredeti *Mosaic*-nál könnyebben kezelhető és megbízhatóbb Netscape Navigator pedig a legelterjedtebb böngésző lett, kiszorítva a *Mosaic*-ot a piacról, mely 1997-ben végleg meg is szűnt.

Az 1994-1995-ben megjelent webböngészők (*IBM Web Explorer, Navipress, SlipKnot, MacWeb, Browse, UdiWWW*) nem veszélyeztették a Netscape Navigator domináns piaci pozícióját, a Microsoft pedig csak az első lépéseket kezdte megtenni a világhálón.

2. (Első) (nagy) böngészőháború

1995-ben a Microsoft megvette a *Spyglass Mosaic* licence-ét, átnevezte Internet Explorer 1-re (az IE 6-ig jelölve volt a böngészőkben, hogy „based on NCSA Mosaic") és a *Windows 95 Plus!* csomagba beépítve, három hónappal később pedig az Internet Explorer 2-t

a Windows felhasználóknak már ingyenesen letölthető formában kezdte forgalmazni (az üzleti ügyfeleknek a Netscape Navigator fizetős, otthonra és oktatási célokra ingyenes volt).

1996-ban a Netscape piaci részesedése még 90% körül volt, de az Internet Explorer 3.0 kezdte funkciók terén beérni a Netscape Navigator-t. A fejlesztési verseny eredménye új funkciók és technikák megjelenése (pl. a Netscape 2.0-ban bevezetett Javascript, ill. az Internet Explorer 3.0-ban bevezetett JScript, néhány egyedi - nem szabványkövető - HTML címke), ugyanakkor a weblapok böngészők szerinti kompatibilitásának romlása, a hibajavítások elmaradása volt.

A döntő fordulat 1997 őszén az Internet Explorer 4.0 megjelenésével következett be. Bár ekkor történt a mai napig használt *Trident* böngészőmotor bevezetése is, az IE-nek a kvázi piaci monopóliummal rendelkező Windows operációs rendszerbe alapértelmezettként való integrálása vezetett az Internet Explorer kiépülő piaci uralmához. 1997 végére a Netscape veszteségbe fordult, 1998-ban elbocsátások kezdődtek és a böngésző forráskódjának nyilvánosságra hozásával a közösségi alapú Mozilla projekt elindításától remélték a Netscape újjászületését. 1999-ben felvásárolta a céget az AOL (America Online) és kivezette a tőzsdéről.

Az USA igazságügyi minisztériuma ugyan monopólium-ellenes eljárást indított 1998-ban a Microsoft ellen a böngészők piacának megszerzése és a versenytársak megsemmisítése céljából az operációs rendszerek terén élvezett erőfölénnyel való visszaélés miatt, de az eljárás csak 2003-ban zárult le. És bár a Microsoftot (akkor már az AOL-nek fizetendő) kártérítésre és 7 éven át az Internet Explorer royalty nélküli használatának engedélyezésére kötelezték, ez már a Netscape-en nem segített. Egyrészt az AOL is IE-t kezdett használni az addigra visszaesett Netscape Navigator helyett, másrészt a programozók nagy részét elbocsátották, a logo-t leszedték az épületről, a folyamatban lévő fejlesztéseket már külsősök végezték. A Netscape ugyan csak 2007 végén szüntette meg (a 9. verzióval) böngészője forgalmazását, akkori piaci részesedése már bőven 1% alatti volt. Ma az AOL egy discount internet szolgáltatója viseli ezt a nevet.

A böngészőháború a Microsoft k.o. győzelmével végződött. 2001 végén - az Internet Explorer 6 megjelenésekor - az IE piaci részesedése 96% volt, meghaladva a Netscape fénykorában elért értéket. Az Internet Explorer-nek egyszerűen nem maradt vetélytársa, ezért megakadt a webböngészők fejlődése is - a következő (IE 7) verzióra 5 évet kellett várni.

3. Új versenyhelyzet kialakulása

A Telenor (akkori nevén Televerk) 1993-ban indította el az első norvég internet szervert és honlapot, és a *Mosaic* helyett egy új böngészőt is fejlesztett *Multitorg Opera* néven. 1995 végén önálló cég - az Opera Software - alakult a böngésző fejlesztésére és forgalmazására, mely kis piaci részesedéssel - főleg Európában – volt használatos, a ’90-es évek második felének böngészőháborújába nem tudott beleszólni. 2003-ban jelent meg az Opera 7. verziója, melyben bevezették a 2013-ig használt, jó CSS és DOM szabványkövető Presto motort, 2005-től (a 8.5 verziótól) pedig a teljes ingyenességet és reklámmentességet.

A német gyökerű KDE 1998-tól fejlesztette a *KHTML* HTML motort és *KJS* Javascript motort, 2000-ben *Konqueror* néven jelent meg a böngészőjük Unix-alapú és Windows operációs rendszerekhez.

A nyílt forráskódú KHTML motort felhasználva indította el 2001-től az Apple a *Webkit* projektet, melyhez a KDE először csatlakozott, majd különvált, de a későbbiekben – többek között - a Nokia, RIM (ma már BlackBerry) és Google is társult.

2003-ban jelent meg a Mac OS X 10.3-hoz a Webkit-re épülő Safari 1 – leváltva az addig használt Internet Explorer for Mac-et - és az Apple a továbbiakban a saját böngészőjét alapértelmezettként integrálta az operációs rendszereibe (a Microsoft 2003-ban le is állította az IE fejlesztését nem Windows-os platformra).

A Mozilla Alapítvány 1998-as létrehozása és 2003 között nem tudott piacképes termékkel előjönni - ekkor jelentették be, hogy *Gecko* motorral önálló böngészőt hoznak létre (először Phoenix, majd Firebird, végül védjegyproblémák miatt) Firefox néven. A Mozilla Firefox 1.0-ot végül 2004. novemberében, az 1.5 verziót 2005. novemberben, a 2.0-t 2006. október végén adták ki.

Az új böngészők megjelenésével 2003-2004-ben elkezdődött az Internet Explorer piaci részesedésének lassú lemorzsolódása, 2007 végére a Microsoft böngésző használata (főleg a Firefox elterjedésének köszönhetően) 77%-ra csökkent.

4. (Második böngészőháború vagy „csak") éles verseny

2008-tól napjainkig tart a böngészők kiélezett versenye, mely a piac jelentős átrendeződését hozta (és melyet szintén szoktak – második - böngészőháborúként nevezni).

2008 márciusában az Apple kiadta Windows-ra a Safari 3.1-et, melyet 2012-ig, az 5.1 verzióig folyamatosan fejlesztett – ekkor felhagyott vele.

2008 júniusban jelent meg a Firefox 3.0, mely a „24 óra alatt legtöbbször letöltött szoftver" kategóriában Guinness-rekordot állított fel – az indulását követő 24 órában 8 millióan töltötték le. A Firefox piaci részesedése egyébként 2010-ig folyamatosan nőtt, utána a Chrome vette át a növekedési dinamikát.

2008 decemberében jelent meg a Chrome 1.0 (csak Windows-ra), a 2010 májusi 5.0 verziótól a Chrome mindhárom platformot támogatja. 2012-re (több, de nem minden mérés szerint) a legelterjedtebb böngészővé vált, 2013-ban (a 28.0 verziótól a Webkit motorról annak egy elágazására, a Blink-re váltott (melyben az Opera is követte a 15.0 verziójától).

Új front nyílt 2007 – 2009 között a mobil (okostelefon és tablet) eszközök terjedésével. A Nokia, Microsoft és RIM/BlackBerry korábbi pozícióit a Google és Apple mobil böngészőinek uralma vette át, míg a Firefox-nak jelentéktelen a mobil piaci részaránya. A Windows Phone 7/8/8.1 IE Mobile 9/10/11-ének jelenlegi egy számjegyű piaci részesedése (becslésenként változó) növekedését prognosztizálják.

5.7. A jelen és belátható jövő (HTML5, CSS3, SVG 2)

A weboldalak struktúráját leíró nyelvet (HTML) egy ember alkotta meg (Tim Berners-Lee, 1989/1991), két szabványosító szervezet dolgozott rajta 5 évig - IETF: HTML 2 (1993-1995) és W3C: HTML 3.2 – HTML 4 (1994-1997) - majd egy évtizedes útkeresés után egy harmadik csoport nyomására (WHATWG) alakul ki a HTML5 szabvány. Széleskörűvé vált a támogatottsága és az új fejlesztések a HTML5 egyre teljesebb implementálására irányulnak, ugyanakkor ez visszafelé kompatibilis a korábbi verziókkal és a régebbi böngészőkkel is használható (mivel azok biztonságosan figyelmen kívül hagyják az új elemeket).

A W3C 2008 eleje után átlagosan négy-öt havonta adott ki HTML5 aktualizált munkatervet (WD) vagy szerkesztői összefoglalót (*Editor's Note*), míg 2012 végére elérte a már elég stabilnak tekinthető ajánlás jelölti (CR) állapotot, mely 2014 decemberére válhat elfogadott ajánlássá (R).

A weblapokat a HTML 4 bevezetése óta a tartalmat és megjelenítést különválasztva célszerű és „szabályos" létrehozni. A weboldalak formázását és megjelenítését szolgáló CSS stíluslapnyelvet a világhálós szabványok „gazdája", a W3C hozta létre és koordinálja a továbbfejlesztését. Évtizedes mozdulatlanság után 2011-ben fogadták el a véglegezett, teljes CSS 2.1 verziót és az első két CSS3 modul (*Színek, Kiválasztók*) végleges ajánlását. A CSS3 funkciói egyre bővülnek, az új modulok az alábbiak (zárójelben a „készültségi fokuk"):

Színek 3 (R)
Kiválasztók 3 (R)
Média lekérdezések (R)
Mozgó tartalom 3 (CR)
Többhasábos elrendezés (CR)
Értékek és mértékegységek 3 (CR)
Kép értékek és helyettesítő tartalmak 3 (CR)
Szövegdíszítés 3 (CR)
Betűk 3 (CR)
Rangsor és öröklődés 3 (CR)
Szintaxis 3 (CR)
Alakzatok (CR)
Irásmódok 3 (CR)
Komponálás és összeolvasztás (CR)
Maszkolás (CR)
Hátterek és szegélyek 3 (WD)
Rugalmas doboz elrendezés (WD)
Átmenetek (WD)
Transzformációk (WD)
Animációk (WD)
Sablon elrendezés (WD)
Háló pozícionálás (WD)
Nyomtatott média 3 (WD)
Tartalomszerkesztés nyomtatott médiához (WD)
Listák és számlálók 3 (WD)
Számláló stílusok 3 (WD)
Előre definiált számláló stílusok (WD)
Kizárások (WD)
Alapvető felhasználói interfészek 3 (WD)
Szűrő effektusok (WD)
Szöveg 3 (WD)
Régiók (WD)
Doboz igazítás 3 (WD)
Túlcsordulás 3 (WD)
Szintaxis 3 (WD)
Belső és külső méretezés (WD)

(WD = Working Draft = munkaterv; CR = Candidate Recommendation = ajánlásra jelölt; R = Recommendation=ajánlás)

A böngészőgyártók – ígéretük szerint - 2014 során új böngészőverzióikból kivezetik a saját előtagok (*vendor-specific prefix*) döntő részét, így a kódolás és megjelenítés ezeknél az új tulajdonságoknál is egységessé, szabványossá válhat. Annál a három modulnál pedig, ahol már elfogadott ajánlásokig jutott el a CSS3 (Kijelölők, Színek és Média lekérdezések), megkezdődött a 4-es szintű modulok kidolgozása (ezek használhatóságának időpontjára korai lenne előrejelzést adni).

Az SVG 2 szabvány elfogadását 2012-ben - a kidolgozása kezdetekor - 2014-re tervezték, ami (részben a kapcsolódó CSS3 modulok elhúzódó véglegezése miatt) nyilvánvalóan csúszást szenved.